



## 저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

# A population-based optimization method using Newton fractal

Soyeong Jeong

Department of Mathematical Sciences  
Graduate School of UNIST

# A population-based optimization method using Newton fractal

A dissertation  
submitted to the Graduate School of UNIST  
in partial fulfillment of the  
requirements for the degree of  
Doctor of Philosophy

Soyeong Jeong

12.13.2018

Approved by

A handwritten signature in black ink, consisting of a large, stylized 'P' followed by a horizontal line and a small loop.

Advisor

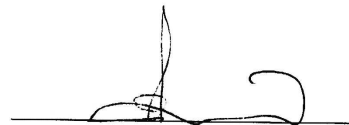
Pilwon Kim

# A population-based optimization method using Newton fractal

Soyeong Jeong

This certifies that the dissertation of Soyeong Jeong is approved.

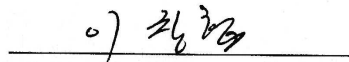
12.13.2018



Advisor: Pilwon Kim



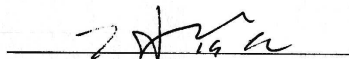
Bongsoo Jang: Thesis Committee Member #1



Chang Hyeong Lee: Thesis Committee Member #2



Jin Hyuk Choi: Thesis Committee Member #3



YunKyong Hyon: Thesis Committee Member #4

## Abstract

Metaheuristic is a general procedure to draw an agreement in a group based on the decision making of each individual beyond heuristic. For last decade, there have been many attempts to develop metaheuristic methods based on swarm intelligence to solve global optimization such as particle swarm optimizer, ant colony optimizer, firefly optimizer. These methods are mostly stochastic and independent on specific problems.

Since metaheuristic methods based on swarm intelligence require no central coordination (or minimal, if any), they are especially well-applicable to those problems which have distributed or parallel structures. Each individual follows few simple rules, keeping the searching cost at a decent level. Despite its simplicity, the methods often yield a fast approximation in good precision, compared to conventional methods.

Exploration and exploitation are two important features that we need to consider to find a global optimum in a high dimensional domain, especially when prior information is not given. Exploration is to investigate the unknown space without using the information from history to find undiscovered optimum. Exploitation is to trace the neighborhood of the current best to improve it using the information from history. Because these two concepts are at opposite ends of spectrum, the tradeoff significantly affects the performance at the limited cost of search.

In this work, we develop a chaos-based metaheuristic method, Newton Particle Optimization(NPO), to solve global optimization problems. The method is based on the Newton method which is a well-established mathematical root-finding procedure. It actively utilizes the chaotic nature of the Newton method to place a proper balance between exploration and exploitation. While most current population-based methods adopt stochastic effects to maximize exploration, they often suffer from weak exploitation. In addition, stochastic methods generally show poor reproducing ability and premature convergence. It has been argued that an alternative approach using chaos may mitigate such disadvantages. The unpredictability of chaos is correspondent with the randomness of stochastic methods. Chaos-based methods are deterministic and therefore easy to reproduce the results with less memory. It has been shown that chaos avoids local optimum better than stochastic methods and buffers the premature convergence issue.

Newton method is deterministic but shows chaotic movements near the roots. It is such complexity that enables the particles to search the space for global optimization. We initialize the particles position randomly at first and choose the leading particles to attract other particles near them. We can make a polynomial function whose roots are those leading particles, called a guiding function. Then we update the positions of particles using the guiding function by Newton method. Since the roots are not updated by Newton method, the leading particles survive after update. For diverse movements of particles, we use modified newton method, which has a coefficient  $m$  in the variation of movements for each particle. Efficiency in local search is closely related to the value of  $m$  which determines the convergence rate of the Newton method. We can control the balance between exploration and exploitation by choice of leading particles.

It is interesting that selection of excellent particles as leading particles not always results in the best result. Including mediocre particles in the roots of guiding function maintains the diversity of particles in position. Though diversity seems to be inefficient at first, those particles contribute to the exploration for global search finally. We study the conditions for the convergence of NPO. NPO enjoys the well-established analysis of the Newton method. This contrasts with other nature-inspired algorithms which have often been criticized for lack of rigorous mathematical ground. We compare the results of NPO with those of two popular metaheuristic methods, particle swarm optimizer(PSO) and firefly optimizer(FO). Though it has been shown that there are no such algorithms superior to all problems by no free lunch theorem, that is why the researchers are concerned about adaptable global optimizer for specific problems. NPO shows good performance to CEC 2013 competition test problems comparing to PSO and FO.



## Contents

<b>List of Figures</b>	<b>ix</b>
<b>List of Tables</b>	<b>xi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Overview . . . . .	1
1.2 Global optimization problems . . . . .	3
1.2.1 Problem settings . . . . .	3
1.2.2 Exploration and exploitation . . . . .	4
1.2.3 Technical issues . . . . .	5
1.3 Summary of contents . . . . .	6
<b>2 Population-based Metaheuristics</b>	<b>8</b>
2.1 Population-based metaheuristics . . . . .	9
2.2 Applications of population-based methods . . . . .	12
<b>3 Exploration and Exploitation in Chaos-based searches</b>	<b>14</b>
3.1 Chaos v.s. randomness in optimization . . . . .	15
3.2 Adoption of chaos in global optimizers . . . . .	15
3.3 Chaos created by Newton method . . . . .	16
<b>4 Newton Particle Optimizer<sup>1</sup></b>	<b>24</b>
4.1 Algorithm . . . . .	24
4.2 Searching manner of NPO . . . . .	26
4.3 Convergence of metaheuristics . . . . .	28
4.4 Local convergence of NPO . . . . .	30
4.5 Criteria for choice of $m$ and $M$ . . . . .	32
<b>5 Construction of A Guiding Function</b>	<b>34</b>
5.1 Conditions for an ideal guiding function . . . . .	34
5.2 Extension of a guiding function . . . . .	35

---

<sup>1</sup>Parts of this work will be published as: Jeong, S. and Kim, P, “A population based optimization method using Newton fractal.” Complexity, *in press*.



---

5.3	Criteria for choice of leading particles . . . . .	37
5.3.1	Idle leaders in leading particles . . . . .	37
5.3.2	Personal best in leading particles . . . . .	38
5.4	Particles outside the boundary . . . . .	39
5.5	Exploration indicator . . . . .	40
<b>6</b>	<b>Results</b>	<b>41</b>
6.1	Tests in 2-dimensional search space . . . . .	41
6.1.1	2-dimensional test functions . . . . .	42
6.1.2	NPO v.s. PSO in 2D . . . . .	42
6.2	Tests in the high dimensional search space . . . . .	43
6.2.1	10-dimensional search space . . . . .	48
6.2.2	30-dimensional search space . . . . .	48
<b>7</b>	<b>Conclusion</b>	<b>53</b>
	<b>References</b>	<b>55</b>

## List of Figures

<b>Figure 3-1</b>	An example of dynamics of a particle under a (guiding) function in the 1-dimensional search space: A particle starting from $-1$ denoted by '1' is searching around a root (boxed point) to find a better approximation. The movement is denoted from '1' to '5'. [74]	18
<b>Figure 3-2</b>	Illustration of the sequences starting from same points. They show different behaviors depending on degree coefficient $m$ for the (guiding) function $f(x) = x(x-1)(x-2)$ [74]	19
<b>Figure 3-3</b>	The basin of attraction of Newton Method shows the Newton fractal. This is an example of Julia sets associated to the Newton method for $f(z) = (z - \mathbf{p}_1)(z - \mathbf{p}_2) \cdots (z - \mathbf{p}_5)$ . The positions of $\mathbf{p}_i$ are denoted by white circles. Each different color region stands for a set of points that converge to the same root by Newton method. Under the same (guiding) function it shows varied fractality depending on degree coefficient $m$ . [74]	21
<b>Figure 3-4</b>	Three Newton paths with different degree coefficient $m$ : They are initiated from the same point $(5, 5)$ . The guiding function is $f(z) = z(z - 2i)(z + 1 - i)$ . [74]	21
<b>Figure 3-5</b>	Newton paths generated from slightly different initial points near $(3, 3)$ (marked with a black circle): both eventually joins again at the target point $(2, -1)$ (marked with a black square) [74].	22
<b>Figure 3-6</b>	The distribution of particles' distance from a leading particle (after 50 times of iterations) follows the power law. All particles are initially located on the unit ball centered at the position of a leading particle. The straight lines are the least squares fitting line for log-log scale depending on $m$ .	23

## LIST OF FIGURES

<b>Figure 4-1</b>	NPO Algorithm. (a)A dotted line is represented the hidden fitness function and the global optimum labeled by diamond shape should be found. (b)Distribute the particles in uniformly random manner. These particles are candidate solution and marked by white circle. (c)Evaluate the fitness of the particles. (d)Choose the top 3 best fitters as leading particles. Leading particles are marked with black square label. (e)Make the guiding function with leading particles. (f)Update the other particle's positions near the best fitter by applying Newton method with guiding function. Then repeat from (c) until it satisfies the stopping criterion. .	27
<b>Figure 4-2</b>	Performance according to $m_{max}$ . The parameter settings are 200 particles with 4 leading particles for rosenbrock function.[74] . . . . .	33
<b>Figure 5-1</b>	A nullcline of a factored guiding function in 2-dimensional space as (4.1.2).	35
<b>Figure 5-2</b>	The distribution of particles using a biased guiding function(above) and a proposed guiding function(below), with same initial condition for the position of particles and degree coefficient $m$ . $t$ counters the iteration step for NPO. $min$ denotes temporal best. . . . .	37
<b>Figure 5-3</b>	The number of leading particles is important. Label denotes the ranking of leading particles among 100 particles. It shows the results of $n = 3$ is better than that of $n = 4$ . . . . .	38
<b>Figure 5-4</b>	Choice of leading particles affects the result: the above adopts top 5 rank performers out of 100 particles as leading particles. The below uses four best fitters and one 40th rank fitter. The blue and red dots indicate the cost values of ordinary and leading particles, respectively.[74] . . . . .	39
<b>Figure 6-1</b>	Test functions which have no local minima. . . . .	44
<b>Figure 6-2</b>	Test functions which have several local minima. . . . .	45
<b>Figure 6-3</b>	Test functions which have many local minima. . . . .	46
<b>Figure 6-4</b>	2-dimensional results of test function in Table 6-1. . . . .	47

## List of Tables

<b>Table 6-1</b>	Test Functions[42]. Each minimum of the functions is 0. . . . .	43
<b>Table 6-2</b>	Benchmark for NPO, PSO, FO: tested with 10-dimensional functions in CEC 2013 competition[74] . . . . .	49
<b>Table 6-3</b>	Performance comparison of NPO, PSO, and FO in the mean ranking[74] .	50
<b>Table 6-4</b>	Benchmark for NPO, PSO, FO: tested with 30-dimensional functions in CEC 2013 competition . . . . .	51
<b>Table 6-5</b>	Performance comparison of NPO, PSO, and FO in the mean ranking . . .	52

# 1

## Introduction

### 1.1 Overview

Groups have different characteristics from each individual. Albeit it is insignificant that a taxi driver has a day off, it is significant that taxi drivers have a day off because it is a strike to express their opinion. To settle some big problems, we gather and exchange ideas. Even though feeble organisms such as ants, birds, fishes in nature, they also go around into a group to achieve their goal. It is surprising that they solve the problem effectively pretending to be an intelligent organism. Swarm intelligence emerges in a scale of groups from nature.

Swarm intelligence has been discussed for last two decades. It is closely related with self-organization, decentralized system, dynamical networks, artificial intelligence, etc. Since the swarm intelligence is not a central-control system, it is adaptable to distributed problem-solving. Their decision making process is a natural selection. Natural selection does not mean that nature selects the fittest but that the decision is made naturally from multi-agents' behavior. This kind of decision process is ideal and reasonable for the equality and fairness of members. Each individual follows some simple rules but their actions result in a powerful and economical collective behavior. In the point of usage for interactions in population it is beneficial to describe the interactive system. This is simulated as a population-based model in computer science. Those population-based models belong to metaheuristic in nature-inspired algorithms[19].

Metaheuristic is a higher-level procedure to take advantage of a heuristic. It is independent on specific problems whereas heuristic is dependent[61]. Population-based metaheuristics are widely used to observe the behavior patterns of people in traffic, trades, etc, or to solve a mathematical optimization problem. Each agent in the swarm is called a 'particle' and has a position in a search space. Each position represents a candidate solution for a global optimization problem. Especially there are many global optimizers derived from a swarm in nature such as Particle swarm optimizer(PSO)[11], Ant colony optimizer(ACO)[8], Firefly optimizer(FO)[65].

For global optimization, the balance between exploration and exploitation is critical. Ex-

Exploration is to investigate the space without using the information from history to prevent the particles from searching near the current best and getting trapped in a local minimum. Exploitation is to trace the neighborhood of the current best to improve it using the information from history. Since these two concepts are at opposite ends of spectrum, the tradeoff among population is the main consideration to reach the global optimum[53, 25].

There are two main categories for global optimization methods. One is deterministic methods, the other is stochastic methods. Classic deterministic methods are well-known gradient descent method, state space search and so on. Typical stochastic methods are the proposed above, metaheuristics using population. These methods are powerful and simple, but they have demerits in difficult replay of implementation, premature convergence, weak exploitation.

Chaos-based optimizer can mitigate these disadvantages. The unpredictability of chaos is correspondent with the randomness of stochastic methods. Because chaos is deterministic, it is easy for a replay with less memory. There are some research papers showing that chaos avoids local optimum better than stochastic methods and buffers the premature convergence issue. There are many trying to use disorder in chaos in conventional stochastic methods and they work well[55, 15, 12, 31, 39]. But stochastic optimization has been studied more often until a recent date because of its simplicity and fast convergence. Stochasticity includes the uncertainty and has a limitation to analyse what is going on in the system. Chaos has different characteristics from randomness of stochastic methods. We utilize the benefit of chaos and make up for the weakness of stochastic methods.

In this dissertation, we propose a population-based metaheuristic, Newton particle optimizer(NPO), which is in press[74]. As suggested from its name, NPO uses the fractality of Newton method in order to update the movements of particle positions in the search space. Though the Newton method is well-known, the chaotic nature of Newton method is rarely known. Newton method is deterministic but shows unpredictable movements in chaos. This complexity of Newton method enables the particles to develop an efficient candidate solution.

The simple procedure for NPO is as follows. Firstly, initialize the position of population in the search space with uniformly distributed random numbers. Secondly, calculate the fitness. And choose some particles to be a candidate for optimum and make them the roots of a polynomial function. These chosen particles are called ‘leading particles’. The polynomial function whose roots are the leading particles drive other particles near them to exploit their neighbourhood by Newton method. It is favorable to choose current best fitters as leading particles but it is tactically more beneficial to maintain the diversity among the leading particles. We achieve the balance between exploration and exploitation by choice of leading particles and the degree coefficient  $m$  in Newton method.

The polynomial function made by these leading particles is called ‘a guiding function’. Then we can apply the Newton method to this guiding function in order to update the positions of particles. Iterate these steps from choosing the adaptable leading particles based on their grades

and diversity to updating the particle position using guiding function. For monotonicity, the current best fitter should include the leading particles. Since the roots are not updated by Newton method, the current best fitter always survives after the update as long as it is chosen as leading particles.

Construction of an ideal guiding function is crucial as well as choosing proper leading particles. It is easy to come up with one in 2 dimensional search space because we have complex number system. We can make such a guiding function satisfying that whose roots are only leading particles and symmetric in dimension and easy to handle. That is nothing but the multiplication of the first order polynomial whose roots are leading particles. But we have to extend the guiding function to higher dimension. The construction rule to multi-dimensional space is discussed in detail later.

We clarify the local convergence of NPO with a modification. Though metaheuristic in nature-inspired optimization has suffered for the difficulties of rigorous mathematical analysis[72], NPO enjoys the strong analysis of Newton method, which has already been studied for a long time. With the results of two popular metaheuristic methods, particle swarm optimizer(PSO) and firefly optimizer(FO), we compare those of NPO. Because there are no such algorithms dominant to all problems by no free lunch theorem[62], the researchers should consider the proper global optimizer for the problems. NPO gives good performance to CEC 2013 competition test problems comparing to PSO and FO.

## 1.2 Global optimization problems

Optimization is one of common tasks that occur in many aspects of a real life. There are many sophisticated optimization tools developed to deal with such optimization problems. We give some examples such as gradient descent method, simplex method, etc. But applying such methods to the problem that we face in our daily life is too expensive and not efficient. We often settle problems by trial and error using our intuition and experience. It is observed that many organisms in nature do in the same way. It is surprising that they sometimes find the nearly optimal solution naturally. We apply this principle into computer simulation for optimization. We make clear some definitions and issues in these following subsections.

### 1.2.1 Problem settings

A function to be optimized,  $f: X \rightarrow Y \subset \mathbb{R}$  is called an objective(cost, fitness) function. The global optimization problem is defined as:

$$\text{find } \vec{x} \in S \subseteq \mathbb{R}^d \text{ satisfying that } f(\vec{x}) \leq f(\vec{y}) \text{ for } \forall \vec{y} \in S, \quad (1.2.1)$$

## 1.2 Global optimization problems

where  $\mathcal{S}$  is the search(parameter) space.  $d$  is the dimension of the problem space.  $\vec{x}$  is the candidate solution in the search space. In this dissertation, we solve so called “the black-box optimization”, which is to find the global minimum of  $f$  with limited prior-information known.

There are some reasons that we prefer not to use conventional methods for this type of optimization problems[67]. Conventional methods usually :

- focus on local search, which is not appropriate for global search.
- require more information on the cost function, such as derivatives.
- are hard to deal with highly nonlinear, multimodal problems and discontinuous functions.
- sensitively depend on starting point.

For these drawbacks in the conventional methods we take approaches from heuristic methods instead. Roughly, ‘heuristic’ means a way of solving problems based on trial-and-errors. It is fast and efficient while trading off between accuracy and precision. In heuristic methods, an agent or a determinant finds the optimum through its own experience and intuition. Heuristics are usually problem-specific[61]. Heuristic approaches can be also computationally expensive and difficult for replay of implementation[67]. The most expensive part of optimizing process is usually calculating the fitness. Most heuristic methods require more iterations to complement the lack of information such as derivative. Heuristic methods include the generation of random numbers.

A metaheuristic is a way of taking advantage of a heuristic to solve general classes of problems, regardless of the specific characteristic of the problems. Metaheuristic methods can be applied to various problems without many confinements[61]. Most metaheuristic algorithms are gradient-free and does not require much information of fitness functions. Metaheuristics do not have to consider nonlinearity, differentiability, or even the continuity, of  $f$ . Though many metaheuristic algorithms include randomness, they are less sensitive than conventional methods or heuristics on starting point because it can remedy the sensitivity using population and escape from the local minima. It can give different solutions even with same staring point like heuristic methods. Thus multiple runs are required with statistical analysis such as mean, variance, median to evaluate the performance.

### 1.2.2 Exploration and exploitation

For an optimal global search, it is essential to make a proper balance between exploration and exploitation. The ideal balance between exploration and exploitation leads us to a global optimum economically. Exploration is to investigate the search space without the information from history, to prevent the particles from getting trapped in a local minimum. Exploitation is to trace the neighbourhood of the temporal best to improve it using the information from history. Not many rigorous studies on the relation between exploration and exploitation has



## 1.2 Global optimization problems

been done yet in spite of its importance. It is difficult to make clear the universality and uncertainty in global optimization problems.

Main issues about exploration and exploitation are following from a perspective of developing an global optimizer[23]:

- definitions of exploration and exploitation.
- whether their relation is the ends of a continuum or orthogonal.
- how they achieve the balance through ambidexterity or punctuated equilibrium.
- which is better, flexibility or professionalism.

Though these concepts are derived from social science, they are worth considering for the practical use in computational optimization. These concepts help us to enrich the contents for optimization. There are some studies like realizing the diverse exploration without sacrificing the exploitation[7] based on a different definition. It is generally accepted that the exploration and exploitation cannot be achieved at the same time but they can sometimes, from the different definition of exploration and exploitation. Another issue is to dealing with problem-specific property. The structure of the given problems affects the manner of the agents' moving during the iterations of optimal processes. And we can divide the process into two parts, exploration and exploitation. Then we assign the tasks for the searching agents. They may have flexibility or professionalism. This affects to the performance.

To guarantee the validity for the optimizer to be introduced, it should be supposed that, at least, the objective function has a decreasing trend near the minimum, on the premise. Because most metaheuristic methods apply the exploitation as putting searching agent near the temporal best. But even in the case that the premise breaks, we still can apply those metaheuristics, since metaheuristic methods are supposed to be independent on problems.

There is so called 'no free lunch' theorem[62] that no algorithm gives the best solution all the time. That is, the average of computation for all optimization problems in the same class is same for all methods. There are no free lunches if the probability distribution on problem instances of solvers is equally distributed. It means no algorithm is dominant all the time : Best optimizer differs across problems. This implies that it is important to use proper metaheuristic method depending on the characteristics of problems.

### 1.2.3 Technical issues

We describe some technical issues involved in the global optimization problems.

#### Initialization

At the initialization step, it is conventional that the positions of particles are uniformly distributed randomly. Though there is no guarantee that random initialization is the best, it is

widely used. Here we use random initialization for all experiments.

#### Stopping criteria

The termination of a global optimizer is a trade-off between efficiency and accuracy. Since it is never guaranteed to find a global optimum, we have to set the stopping criteria. We expect that the more iterations the optimizer did, the better the result would be as long as the population avoid the local minimum successfully. But as iterations progress, it is much harder to improve the current best. From the perspective of reliability and efficiency, setting proper stopping criteria is important.

Limitation of the number of fitness evaluations is one of the most popular options. Evaluation of fitness is usually a computationally-expensive part. Thus it is better with less number of fitness evaluations though two experiments result in the same minimum values. Usually, the adaptable number of fitness evaluation is the multiplication of 10000 and the dimension of the search space[42]. One of the widely known problem sets, CEC 2013 problem set is also chosen that. Here we use this criteria.

For other options, there are various performance metric such as generational distance[71], epsilon indicators[60], density[50]. These various metrics are adaptable for more difficult problems. Because the accuracy limited to the number of fitness evaluations is insufficient for converging to the global minimum[52]. There has been not much research done on stopping criteria for various problems yet.

#### Performance evaluation

Due to the stochasticity, the performances are usually evaluated in statistical sense. Multiple runs are required to validate the experimental results and we did 51 runs as suggested in CEC 2013 problems set[42]. To get the median, an odd number is chosen. We tried to repeat the process more than 51 times but the results seemed not to converge as the iteration number increased because of the complex structure with limited computer memory.

Usually, the average of cost of current best for multiple runs is an important factor. An global optimizer is said to be better if it has higher probability to the reach the better result for multiple runs.

### 1.3 Summary of contents

In chapter 2, we introduce some backgrounds to understand the usage of swarm intelligence. In §2.1, some typical population-based metaheuristics are introduced, such as particle swarm optimizer, ant colony optimizer, firefly optimizer. And the applications of swarm intelligence

optimization are discussed in §2.2.

In chapter 3, we consider the trade-off between exploration and exploitation using chaos. This chapter explains how chaos is used for global optimization problems. In the first section, the difference of chaos and randomness is introduced. Metaheuristics introduced in the previous chapter can be modified using chaos instead of generation of random numbers. Because chaos can be controlled with fewer variables and is well-structured, this property is effective for global optimization. The detailed is written in §3.2. Finally, we check the chaotic nature of Newton method to make a link between Newton method and global optimization problems.

In chapter 4, we propose the Newton particle optimizer(NPO)[74]. We elaborate on the algorithmic procedure of NPO in the first section. And searching manner of NPO is in the following section comparing with other conventional methods. In §4.3 the convergence issues are written mainly based on the premature convergence, local convergence, and global convergence. In §4.4 guaranteed convergence NPO is suggested for local convergence. In the last section, the control of parameters, degree of coefficient  $m$ , is suggested for the better performance.

In chapter 5, we discuss the construction of an ideal guiding function for NPO. The extension of a guiding function in multi-dimensional space is the main issue to apply NPO in the dimension beyond 2-dimensional space. This chapter contributes to the improvement of the performance of NPO. We introduce some practical issues for the choice of leading particles in §5.3. Dealing with particles outside the searching domain is discussed in §5.4. And we suggest the exploration indicator to measure the ratio between exploration and exploitation in NPO in §5.5.

In chapter 6, we apply NPO to the real problem set. We introduce some test functions and show the 2-dimensional results in §6.2. We compare the results with PSO and FO in 10 and 30 dimensional space using CEC 2013 problem set, which consists of widely renowned benchmark functions for global optimization.

In chapter 7, the conclusion is discussed.

## 2

# Population-based Metaheuristics

Population-based optimization algorithms mimic the swarm intelligence in nature basically. We see the swarm intelligence in our daily life, such as bird flocking, fish schooling, ant colonies or human social behaviours. Closely observing these collective behaviours, many researchers came up with ideas about population-based optimizations. Popular examples are particle swarm optimization, firefly optimization, ant colony optimization, and so on.

Collective behaviours are interesting in the way that the swarm intelligence is decentralized and self-organized, without any controller to govern the whole phenomena or to give any order to each agent. Each individual in a swarm follows some simple rules, which results in the collective behaviour in macro scale. From the outside, it seems that the swarm is an intelligent agent but the swarm has a system to achieve their goal based on their implicit patterns of behaviours. We are interested in how the system works because it is effective or close to be optimal. The system is working in the way to take best advantages of the limited resources. That is why they survive for last thousands of times in the nature.

We use these kinds of systems as metaheuristics in computer science and mathematical optimization. The term ‘metaheuristic’ was proposed by Glover[19]. Each agent in a swarm behaves by trial and error based on its experience, that is, in a heuristic way. A metaheuristic is literally a higher level of procedure beyond heuristic to utilise the information from heuristic. A heuristic is dependent on the specific characteristic of problems, whereas a metaheuristic is independent. Metaheuristics can be applied when we have little information. We call those kinds of problems black box problems. It is difficult to solve black box problem with conventional methods such as gradient descent method. We introduce some nature-inspired metaheuristic methods, especially population-based optimization algorithms in this chapter.

Search through the interaction of multiagent is more beneficial than the sum of information of each agent’s search. An agent inspects the search space using its experience and makes a decision for the next step. The strategy for the next step is based on simple rules with a little uncertainty. However the experience for an agent is often not enough to make a decision in

complex life. And the data is likely too big and complex to analyse. So multiagent can search the space more stably. If they interact each other, multiagent move with more information though it is computationally heavy. Since the computation of fitness is usually much more expensive than data processing, population-based methods are favorable.

## 2.1 Population-based metaheuristics

We introduce four conventional population-based metaheuristics which have been widely used in practical applications. The proper population size should be ‘not as large as to be dealt with statistical averages’ and ‘not as small as to be dealt with as a few-body problem’[3].

### Particle swarm optimization

Particle swarm optimization(PSO) has been proposed by Kennedy and Eberhart in 1995, in the effect of imitating swarm behaviours of bird flocking and fish schooling[11]. Because PSO is simple and convergent fastly, it is one of the most popular metaheuristics. PSO is juggling with these following 4 main vectors to update the positions of particles[73].

- *Position*( $x_t^i$ ) : The position vector of the  $i$ -th particle at iteration  $t$  step.
- *Velocity*( $v_t^i$ ) : The velocity vector of the  $i$ -th particle at iteration  $t$  step.
- *PersonalBest*( $p_t^i$ ) : The best position of  $i$ -th particle from all iterations in history.
- *GlobalBest*( $g_t$ ) : The best position of all the particles at iteration  $t$  step.

With these 4 vectors, PSO updates the particle’s position as follows.

$$\begin{aligned}
v_{t+1}^i &= \omega v_t^i + \phi_p r_p (p_t^i - x_t^i) + \phi_g r_g (g_t - x_t^i) \\
x_{t+1}^i &= x_t^i + v_{t+1}^i \\
p_{t+1}^i &= \begin{cases} x_{t+1}^i & \text{if } f(x_{t+1}^i) < f(p_t^i) \\ p_t^i & \text{otherwise} \end{cases} \\
g_t &= x_t^j \quad \text{where } f(x_t^j) \leq f(x_t^i) \quad \text{for } i \in P.
\end{aligned}$$

where  $P$  is the population set.  $r_p, r_g$  are uniformly distributed random numbers in  $[0, 1]$ . Parameter  $\omega, \phi_p, \phi_g$  are the inertia weight, cognitive weight, social weight, respectively[73]. These three weights are parameters to control the update of velocity. Inertia weight  $\omega$  was introduced in 1998. If  $\omega$  is 1, then it is called Original PSO(OPSO). Otherwise it is called Standard PSO(SPSO). Cognitive weight  $\phi_p$  and social weight  $\phi_g$  are sometimes called acceleration coefficients. The first term of velocity( $v_t^i$ ) update rule is reflected the impact of the previous velocity controlled by  $\omega$ . The second term is called cognitive impact, and the third is social impact.

Since the personal best is replaced once they find the better particles from each particle's history, this has something to do with cognitive skill of individual. Moreover the global best is found through the interaction of all particles. This term relates with the social skill of a swarm.

The convergence of PSO has been studied in many ways[44]. There has been done many researches about the convergence of PSO using constriction coefficient[6], limit[36], differential equation[69], matrix[57, 35], difference equation[17], Z transformation[32], etc. We show the main convergence result using limit.

*Definition 2.1.1.* (Variance of population's fitness)[36] The variance of the population's fitness,  $\sigma^2$ , is defined as

$$\sigma^2 = \sum_{i=1}^n \left( \frac{f_i - \text{avg}(f)}{f} \right)^2$$

where  $n$  is the number of particles,  $f_i$  is the fitness value of particle  $i$ ,  $\text{avg}(f)$  means the average value of all fitness of the swarm,  $f$  is the normalizing factor to restrict  $\sigma^2$ .

This following theorem shows the relation between the convergence and fitness values.

**Theorem 2.1.1.** [36] *If PSO algorithm is prematurely convergent or global convergent, the particles converge to one or some places in the search space  $S$ , and  $\sigma^2 = 0$ .*

We say PSO tends to lose diversity as the iteration goes on. As a matter of fact, most metaheuristic suffers from this phenomenon before the population finds the optimum. We call this convergence premature convergence.

## Ant colony optimization

Ant colony optimization(ACO) has been developed by Marco Dorigo in 1992[8]. This algorithm is the first attempt to establish a link between the behaviour of ants and computer science. All of the individual ants communicate each other locally with 'pheromone'. This chemical substance, pheromone, carries out as a messenger by deposition and evaporation. To survive in the barren land, it is essential for ants to find the shortest path connecting the ants' nest and food. No ants control the colony and give an order to each agent.

But it is interesting that they find the shortest way in the end. It is a good example of problem-distributed solving and natural selection in nature. Each ant just follow the simple rule. They lay down the pheromone in the way they pass at random initially. Then some follower ants tend to choose strong pheromone trail. If the path is shorter, the more ants go and come more often for the same time. It means that it is naturally selected for a shorter path to have a strong pheromone by ants. As time goes on, the pheromone will be evaporated for a path no ants choose. On the other hand, the strong pheromone trail attracts more ants resulting in stronger pheromone path. This positive feedback system can be taken advantage

of combinatorial and continuous optimization problems.

### Firefly optimization

Firefly optimization(FO) has been proposed by Xin-She Yang in 2008 mimicking the pattern of light-flashing of fireflies[65]. The fireflies follow these three rules. First, all fireflies are unisexual. Any individual firefly will be attracted to all other fireflies. Second, attractiveness is proportional to their brightness. For any two fireflies, the less bright one will be attracted by the brighter one and move towards that. The brightness intensity decrease as their distance increases. Third, the fireflies will move randomly if no other firefly is brighter than them.

The light absorption decays exponentially and light variation follows the inverse square law by distance. FO updates the position as follows.

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{ij}^2} (x_j^t - x_i^t) + \alpha \epsilon_i^t$$

where  $\alpha$  is the scaling factor for controlling the step sizes of the random numbers.  $\gamma$  is a scale-dependent parameter for controlling the visibility of the fireflies.  $\beta_0$  is the attractive constant.

It is worthy of comparing FO and PSO[66]. Note that as  $\gamma$ , in FO, converges to 0, that would be the standard PSO. Computationally, FO is nonlinear whereas PSO is linear. Thus we enjoy more dynamic nature with FO. Actually, FO can implement the multi-swarms with strong nonlinearity. This implies that FO can be more efficiently applied to solve multimodal problems. And we do not have to consider the initialization of velocity with FO, unlike PSO, because there is no velocity terms in FO. On the other hand, FO has a scaling factor,  $\gamma$ , which is used for various problems with less manipulation.

### Genetic algorithm

Genetic algorithm(GA) has been developed by John Holland in 1975[27]. It simulates based on evolution theory of Darwin. The main operators are ‘crossover, mutation, selection’. Through the generation of populations, the operators are applied probabilistically. It is hard to apply for design problems because the mutation often happens in a large search space. So it should be taken apart into the simple representation. Like other metaheuristics, GA tends to stuck in the local minima. One way to avoid this phenomenon is maintaining the diversity. For GA, the importance of diversity is bigger than other methods because the crossover of a homogeneous population is not proper for the generation of new solutions. One way is imposing some disadvantages for some similar populations but it is inefficient and dependent on the specific problem.

## 2.2 Applications of population-based methods

Here we present several applications which population-based optimizations show outstanding performances in[45].

### Clustering and classification

Clustering and classification are fundamental techniques for machine learning and data mining. For clustering the input data, each particle in an optimizer is a cluster centroid vector[59]. Population represents candidate clusterings for the data set. Then the fitness of particles is as follows.

$$fitness := \frac{\sum_{j=1}^{N_c} [\sum_{z_p \in C_{ij}} d(z_p, m_j) / |C_{ij}|]}{N_c}$$

$$d(z_p, m_j) := \sqrt{\sum_{k=1}^{N_d} (z_{pk} - m_{jk})^2}$$

where  $z_p$  is p-th data vector and  $m_j$  denotes the centroid vector of cluster  $j$ . Subscript  $k$  is the dimension. Here  $N_d$  is the input dimension, and  $N_c$  is the number of cluster centroids.  $|C_{ij}|$  is the number of data vectors belonging to cluster  $C_{ij}$ . PSO shows a better convergence with lower fitness errors[2, 29].

### Control

There are many optimization-related issues in a wide range of control problems such as adaptive control, fuzzy control, proportional-integral-derivative(PID) control. Especially in handling the difficulties from high order, time delays, and nonlinearities, heuristic methods yield great performances[16]. Many artificial intelligence (AI) techniques such as neural network, fuzzy system, and neural-fuzzy logic have been taken to tune controller parameters. Especially genetic algorithm has been successfully applied to such problems. However, due to the lack of diversity and the premature convergence, it deteriorates numerical performance. We can apply any other metaheuristics to overcome such difficulties[24, 22, 1].

### Image and video

Maintaining the high resolution with as small capacity as possible in the image and video is the important issue. As watching video is on-trend, the traffic for video has been increased so rapidly. It needs to be found how much bitrate should be assigned with limited resources on



## 2.2 Applications of population-based methods

---

a video. Within the framework of optimization problems, each particle represents the combination of resolution and bitrate with a constraint. Because the feature is changing in a video, the adaptable combination keeps changing. This mechanism is similar to an evolutionary process against constantly changing environment in nature.

### Neural network training

Artificial neural networks are being used to process the input data in many areas, such as classification, feature extraction, clustering, and approximate inference. A neural network function is defined as a composition of weighted sum of nonlinear activation functions, mathematically. Hyperbolic tangent, sigmoid or rectifier function are popular for an activation function.

Training a neural network is an optimization problem to find the weight of a neural network function that minimizes an error function. From the perspective of global optimization problems, it is desirable that a smooth change of the neural network function's weight results in a smooth change of the value of an error function. In this case, gradient descent method works well. But if an activation function is non-differentiable, or complicated, population-based metaheuristics are adaptable.[13, 47] Population-based metaheuristics does not require much information about the cost function. In addition, evolutionary algorithms(a subset population-based metaheuristics) can be used to optimize network structure as well as network parameters in combination sense.[10] In this permutation problem, due to the topological symmetry in the error function, there exists many local minima.

### 3

## Exploration and Exploitation in Chaos-based searches

We introduce the concept of chaotic-based search and its nature, especially in terms of balancing between exploration and exploitation in this chapter. Chaos theory is a theory of nonlinear dynamical systems that deals with interaction between order and disorder. Though a chaotic system may follow few simple rules, it can generate irregular dynamics in a bounded region. Such characteristic of chaos can be applied to find an optimal solution. Chaos-based search has these three important properties[64].

- the sensitive dependence on initial conditions
- the semi-stochastic property
- ergodicity

Even with a slightly different initial condition, the behaviour of chaotic system is totally different. This attribute is distinct from the predictability of linear systems. Strictly speaking, chaos is predictable as long as we know the exact information of the state, which is almost impossible. As of now, chaos is extremely hard to predict due to the limitation of observation equipments. Randomness of stochasticity is a disorder, whereas chaos is complex like randomness but well-structured. That is why we say chaos has the semi-stochastic property. This property can be used for optimization algorithm instead of randomness. Though chaos is sensitive on initial condition, the chaos based optimizer is quite stable on initial condition because of the third property of chaos, ergodicity.

Ergodicity has something to do with the uniformly distribution of random numbers with respect to time and space. Ergodicity means the time average of random process is the same as its average over the probability space. This enables the optimizer to inspect the space meticulously in search space. Since chaos has an ergodic property and is densely periodic, chaotic based optimizer finds the optimum stably regardless of the initial condition.

In following sections we will see using chaos is reasonable for global optimization problem enabling us to tune the balance between exploration and exploitation effectively.

### 3.1 Chaos v.s. randomness in optimization

A deterministic system consists of three followings ingredients[26]:

- the time-evolution equations
- the values of the parameters describing the system
- the initial condition.

Deterministic optimizers such as gradient descent method are good at finding local minima in convex problems. But they often fail to find a global minimum especially in non-convex and complex problems. By using chaos we enjoy not only the advantages of deterministic system, but also semi-randomness which is similar to stochastic system. So chaos sometimes is said to be ‘semi-deterministic’[49] or pseudo random[9].

The characteristics of a deterministic system make it more beneficial to use chaos instead of randomness in global optimizer. Usually global optimizer has many iterations. So stochastic optimizer needs many random numbers and computational memory to repeat the implementation. But we can get the semi-random numbers from chaos as time goes by. What is more, these random numbers can be somewhat predictable within a few steps and are sensitive on initial condition with a few degrees of freedom because it has only a few parameters in the system. A degree of freedom is the number of independent variables to describe the system[26]. Comparing with the fact that randomness has so many degrees of freedom, the random numbers from chaos saves us the computational memory and cost.

Strictly speaking, chaos is hard to predict while random noise is completely unpredictable. Being hard to predict is different from being unpredictable. Though the behaviour of chaos is very hard to predict, it is already determined depending on the initial condition and a few parameters. The more information of the system we have, the better the prediction of behaviour would be improved. We can gather more evidence to guess. But for stochastic system, we cannot know the result even if we have all the exact information of the current state. No one knows the result until it has been done though it has same initial condition. The nature of stochasticity involves uncertainty, which is not desired.

### 3.2 Adoption of chaos in global optimizers

This section contributes to check the characteristics of chaos for global optimizers. There are many conventional optimizers such as PSO, FO, GA using chaos[55, 15, 12, 31, 39]. They show the chaotic nature is effective for global optimization. There are two main advantages in the following.

### Strong exploitation

Chaos system is aperiodic which means that the state is never repeated no matter how much time passes[26]. Actually chaos has a dense periodic orbit. This enables the particles to exploit near temporal best candidate solution. This is a desired phenomenon as it has been shown that the particles' state space of PSO is not a recurrent process[48].

Though chaos is hard to predict, it is bounded in a region for a few initial steps. Using this property we can enjoy exploitation by control of the number of iterations. Lower number of iterations enables exploitation whereas higher number enables exploration in the bounded domain.

### Avoidance from local minima

Since most population-based metaheuristics move particles by the interaction among population, this finally results in the premature convergence to some places in the search space[43]. Particles' stagnation mostly happens before they find the global minima because of lack of diversity within the population[37]. However, the chaos-based optimizers drive particles using chaotic nature regardless of the interaction by accident. Chaos makes the continuous search in a wide area, say, exploration. Chaos is topological mixing and this property plays a role like uniformly distributed random numbers. The particles driven by chaos never gather, and never stop.

## 3.3 Chaos created by Newton method

Newton method is widely renowned but not many people know that Newton method has a chaotic nature. We introduce the chaos created by Newton method in this section.

Newton's (or the Newton-Raphson) method[28, 51] is one of the most popular iteration methods to find roots of the continuously nonlinear differentiable functions where

$$x : f(x) = 0.$$

It linearly approximates one of the roots depending on initial guess  $x_0$  according to this recurrent process

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

as  $n$  gets a larger integer. It can be proved in many ways. Assume that  $f \in C^2[a, b]$ , where  $x_0 \in [a, b]$  is an approximation to the solution  $x$  of  $f(x) = 0$  such that  $f'(x_0)$  is non-zero, and

### 3.3 Chaos created by Newton method

$|x - x_0|$  is small. By Taylor series,

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \frac{(x - x_0)^2}{2}f''(\xi(x))$$

where  $\xi \in (x, x_0)$ . Higher order terms are small and negligible. Since we are looking for  $x$  satisfying that  $f(x) = 0$ ,

$$0 = f(x_0) + (x - x_0)f'(x_0).$$

Then,

$$x = x_0 - \frac{f(x_0)}{f'(x_0)} := x_1.$$

For diverse movements, we apply modified newton method, which includes a degree coefficient  $m$ . For a nonlinear function  $f$ , the approximations to the roots are updated as

$$x \longleftarrow x - m \frac{f(x)}{f'(x)}. \quad (3.3.1)$$

If we choose  $x$  close to one of its roots of  $f$  initially, it fastly converges to the corresponding roots. The sequence generated from (3.3.1) can be susceptible to the starting point and the degree coefficient  $m$ . We can check the sensitivity through visulatzation of basin of attraction in complex system. If  $f : \mathbb{C} \rightarrow \mathbb{C}$  is a complex function then the basin of attraction of Newton method shows a fractal nature[38]. The trace of  $x$  is called *Newton path*.

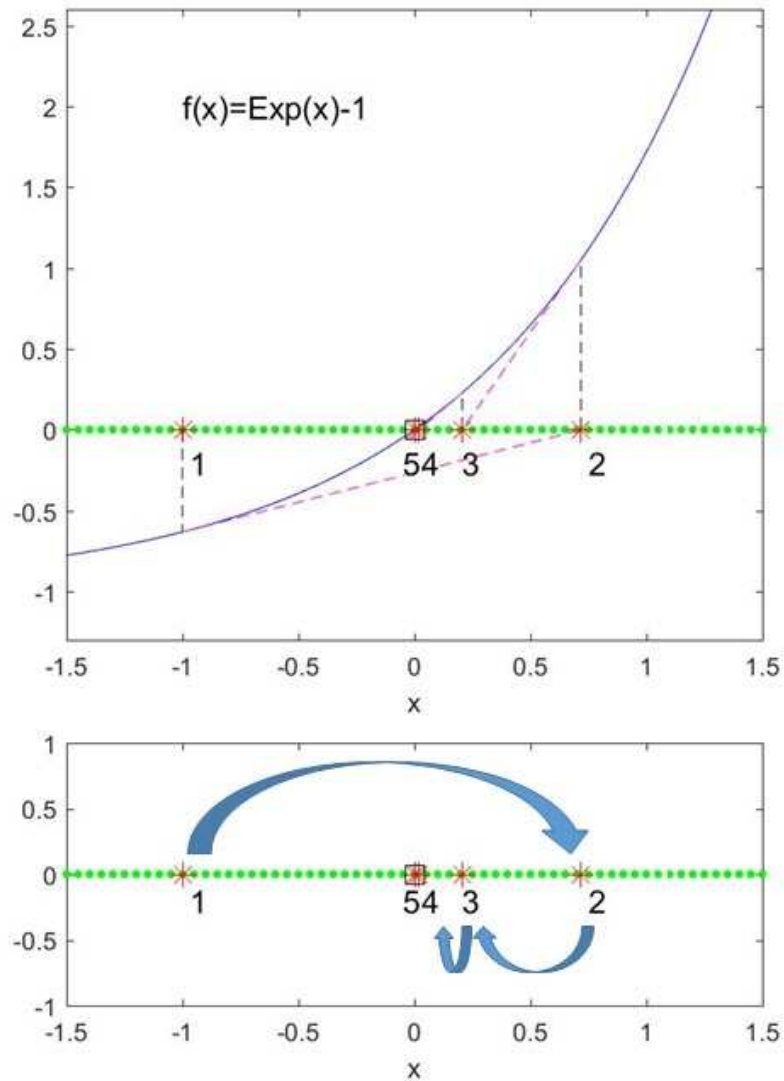
A polynomial function whose roots are  $p_1, p_2, \dots, p_n$  can be constructed as

$$f(x) = (x - p_1)(x - p_2) \cdots (x - p_n) \quad (3.3.2)$$

where  $p_i$  are  $n$  locations of interest in  $\mathbb{R}$ . If we apply the Newton method (3.3.1) to (3.3.2) iteratively, the particle  $x$  wanders around those roots. We check that in Figure 3-1. We call this function as ‘guiding function’ as in (4.1.1), introduced later.

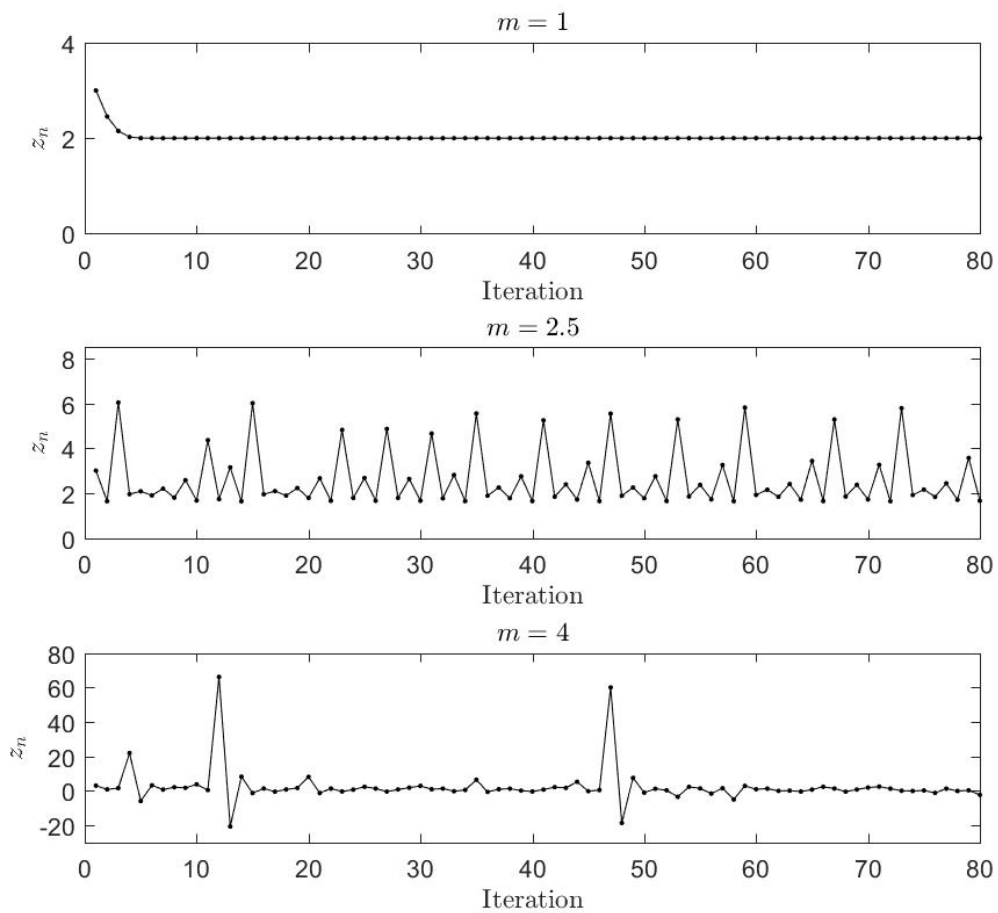
The behaviour of a point is sensitive depending on the choice of  $m$ . As in Figure 3-2 the sequence of a particle whose degree coefficient  $m$  is 1 converges so fast to one of the roots,  $x = 2$ , in (a), whereas the sequences of  $m = 2.5$  and  $m = 4$  in (b) and (c), respectively, seem to wander around the roots,  $x = 0, 1$ , or  $2$ , during the iterations. The movement of a particle whose degree coefficient  $m$  is a higher value is more irregular around the root, even jumping around 60. However, it is an important observation that the movements are bounded and never escape from  $x = 2$ .

In the 2-dimensional space the dynamics is even more diversified under the (guiding) function,  $f = f(z), z \in \mathbb{C}$ . Sensitivity on the initial point, of Newton method, divides the domain into complicated regions. We call them *Julia sets*, according to the points where the elements of the each region converge to. In Figure 3-3, each region with a different color stands for a set



**Figure 3-1:** An example of dynamics of a particle under a (guiding) function in the 1-dimensional search space: A particle starting from  $-1$  denoted by '1' is searching around a root (boxed point) to find a better approximation. The movement is denoted from '1' to '5'.[74]

### 3.3 Chaos created by Newton method



**Figure 3-2:** Illustration of the sequences starting from same points. They show different behaviors depending on degree coefficient  $m$  for the (guiding) function  $f(x) = x(x-1)(x-2)$ [74]

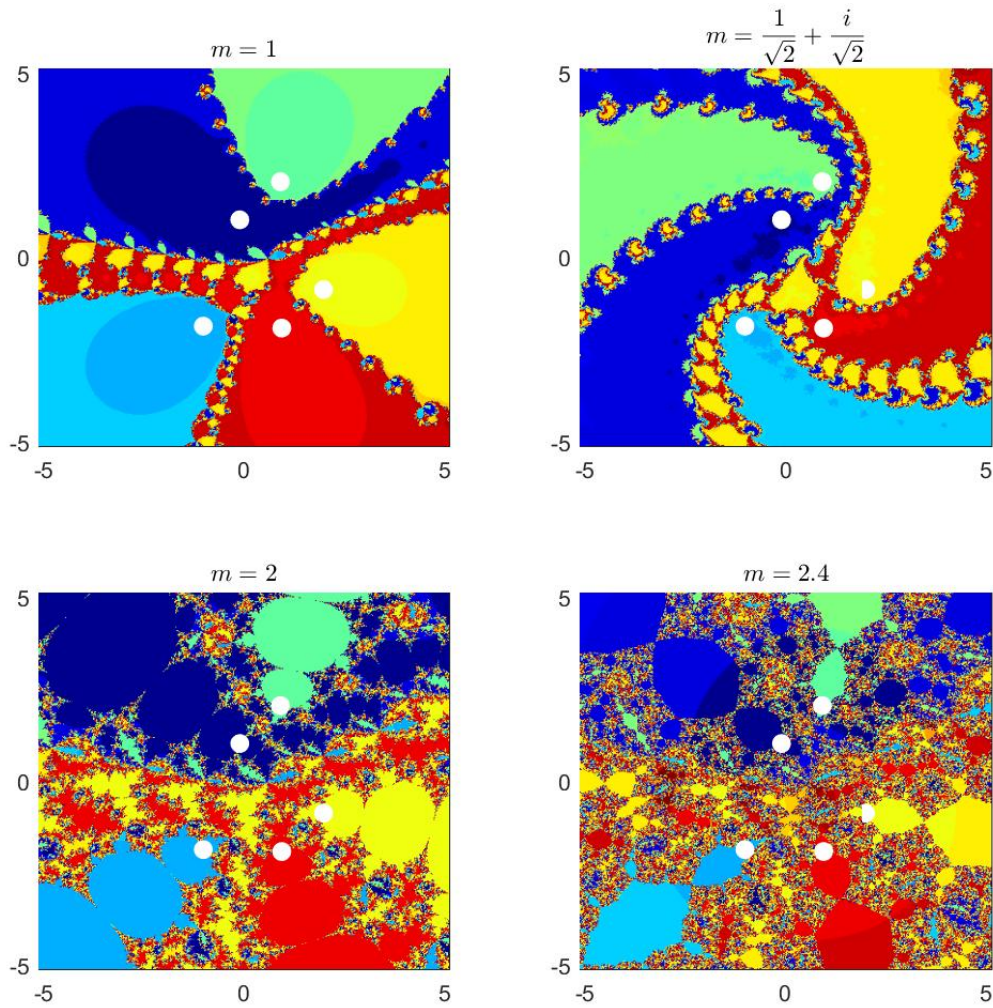
### 3.3 Chaos created by Newton method

of points that converge to the same root by Newton method. The boundaries of such regions show the fractal geometry, implying the corresponding Newton paths are very diverse.

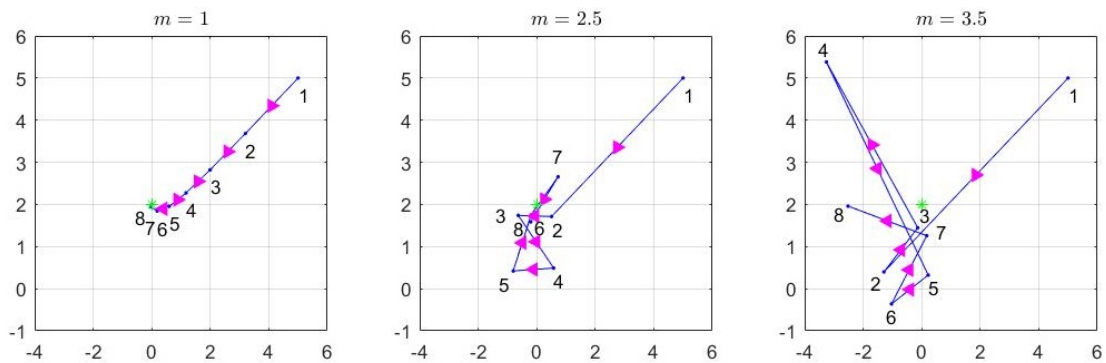
We compare three Newton paths in Figure 3-4 with distinct values of  $m$  in the complex plane. Three examples show the movement of particles initiated at the same point  $(5, 5)$ . Those are attracted to the origin under the same (guiding) function  $f(z) = z(z - 2i)(z + 1 - i)$ . The particle with  $m = 1$  make a gradual search toward the origin, whereas the ones with  $m = 2.5$  and  $m = 3.5$  show rather irregular motions around it. It is remarkable that the movements of the latter cases are erratic mix of jumping and mincing. Figure 3-5 shows the sensitivity on the initial points. The Newton paths are totally different though we start from the slightly different initial points near  $(3, 3)$ . They converge to the same point in the end. This “diversely- convergent” searching paths enables the population to search the space balancing between exploration and exploitation.

It is notable that the particles jump depending on long tail distribution as in Figure 3-6. The distribution covers a wider area with  $m$  increased, indicating stronger exploration. However, the presence of the power-law implies that the particles exploit as well as explore simultaneously.



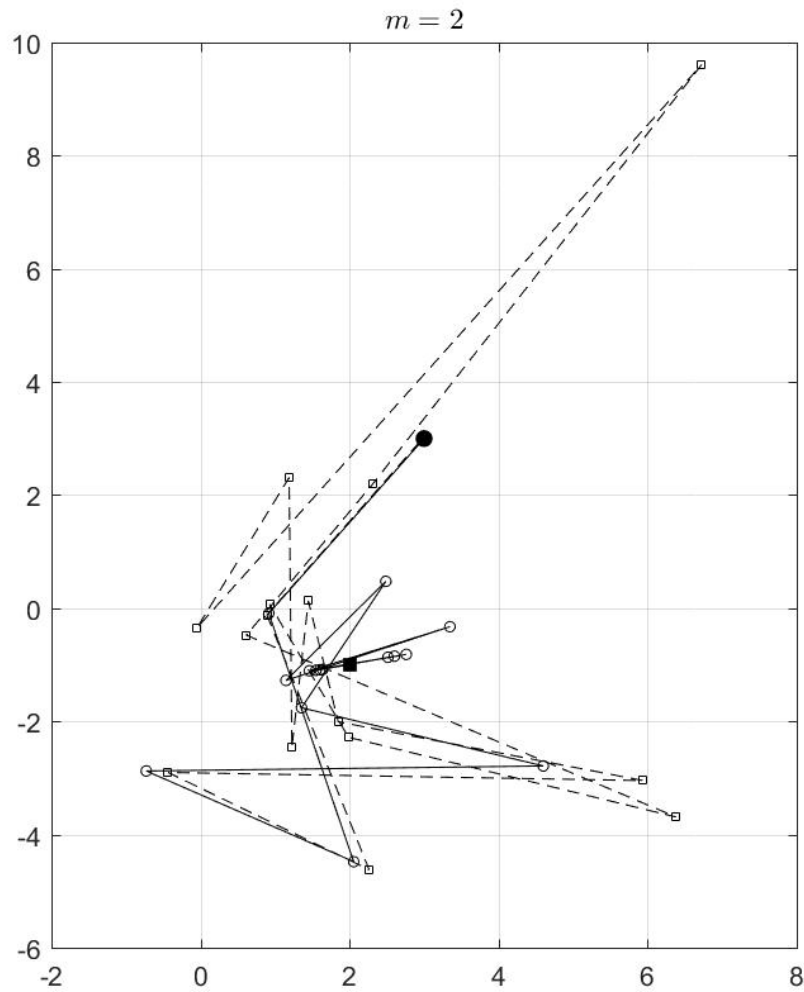


**Figure 3-3:** The basin of attraction of Newton Method shows the Newton fractal. This is an example of Julia sets associated to the Newton method for  $f(z) = (z - \mathbf{p}_1)(z - \mathbf{p}_2) \cdots (z - \mathbf{p}_5)$ . The positions of  $\mathbf{p}_i$  are denoted by white circles. Each different color region stands for a set of points that converge to the same root by Newton method. Under the same (guiding) function it shows varied fractality depending on degree coefficient  $m$ . [74]

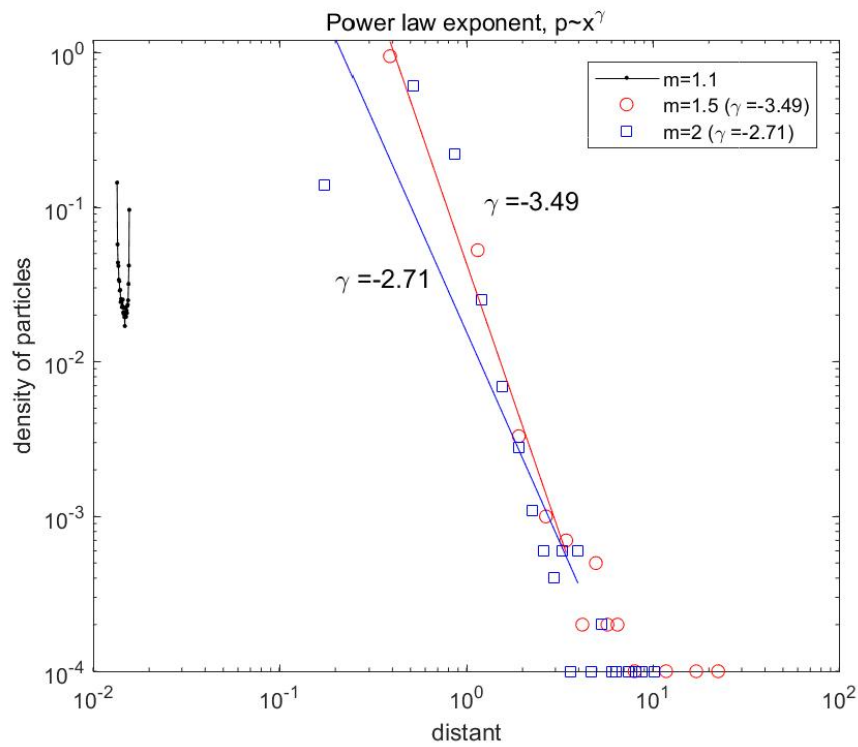


**Figure 3-4:** Three Newton paths with different degree coefficient  $m$ : They are initiated from the same point  $(5, 5)$ . The guiding function is  $f(z) = z(z - 2i)(z + 1 - i)$ . [74]

### 3.3 Chaos created by Newton method



**Figure 3-5:** Newton paths generated from slightly different initial points near  $(3, 3)$  (marked with a black circle): both eventually joins again at the target point  $(2, -1)$  (marked with a black square)[74].



**Figure 3-6:** The distribution of particles' distance from a leading particle(after 50 times of iterations) follows the power law. All particles are initially located on the unit ball centered at the position of a leading particle. The straight lines are the least squares fitting line for log-log scale depending on  $m$ .

## 4

# Newton Particle Optimizer<sup>1</sup>

We propose Newton particle optimization(NPO) algorithm[74] and its convergence in this chapter. NPO is a population-based metaheuristic for global optimization problems. NPO enables the particles to search the space balancing between exploitation and exploration based on the chaotic nature of Newton method. Using chaos, NPO can control the irregular behavior with a few degree of freedom. This implies that NPO requires less random numbers than other stochastic metaheuristic methods. NPO is simple and powerful. In addition NPO is convergent fastly due to the convergence property of Newton method.

Complexity of the Newton paths can used to develop a global optimizer. It can be constructed for a guiding function  $f$  whose roots are the temporal best, or candidate for optimums, of population according to fitness function  $g$ . By (3.3.1) all other particles are attracted to the temporal best of  $g$  along the Newton paths. The sequences of particles' positions are irregular, likely to wander around the temporal best, or sometimes take an unexpected jump away from them. If the temporal best is improved, the guiding function  $f$  is updated accordingly. We update the position of particles with newly constructed guiding function whose roots are temporal best by Newton method at every iteration step.

## 4.1 Algorithm

We explain the algorithm of NPO in 2-dimensional search space first. A fitness function(or cost function)  $g : \mathbb{C} \rightarrow \mathbb{R}$  is a function to be optimized. Each particle is a searching agent and the population is  $N$ . Their positions are denoted as  $z_i \in \mathbb{C}$ ,  $1 \leq i \leq N$ . Each particle has a degree coefficient  $m_i \in \mathbb{C}$ . By Newton method the position of  $i$ -th particle  $z_i$  is updated as following:

$$z_i \leftarrow z_i - m_i \frac{f(z_i)}{f'(z_i)} \quad (4.1.1)$$

---

<sup>1</sup>This work will be published as: Jeong, S. and Kim, P, "A population based optimization method using Newton fractal." Complexity, *in press*.

#### 4.1 Algorithm

where  $f$  is a polynomial function that is defined to be a *guiding function*. Do not confuse  $f$  with  $g$ , to be optimized. We can attract the particles near its roots by (4.1.1). That is why we need a root finding method like Newton method. The main concern is about how to construct the guiding function  $f$  to apply (4.1.1) for all particles.

We easily come up with a guiding function in 2-dimensional space because of the complex system. A guiding function  $f(z)$  is constructed as a polynomial of degree  $n$  whose roots coincide with  $\mathbf{p}_1, \dots, \mathbf{p}_n$ , as

$$f(z) = (z - \mathbf{p}_1)(z - \mathbf{p}_2) \cdots (z - \mathbf{p}_n). \quad (4.1.2)$$

We can use the guiding function to search the space around its roots, possibly temporal best solutions, which are the candidate optimum.

At each iteration of the scheme, we choose  $n$  best fitters with respect to the fitness function  $g$  as *leading particles*. That is, we pick  $\mathbf{p}_1, \dots, \mathbf{p}_n$ , such that

$$\mathbf{p}_i = z_k \text{ where } g(z_k) \text{ is the } i\text{-th minimum among } g(z_1) \cdots, g(z_N) \quad (4.1.3)$$

for  $i = 1, \dots, n$ . It always needs not  $n$  best fitters to be leading particles but the first best fitter should belong to the leading particles for monotonicity to the optimum.

Note that all the particles except the leading particles are attracted toward leading particles. Once the positions of all particles are updated, we examine their fitness to choose the next leading particles  $\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n$ . Then we refresh the guiding function  $f$  accordingly, and reapply (4.1.1) to the particles. As long as the guiding function is a polynomial function, convergence of the algorithm is guaranteed from the convergence of the Newton's method and the monotone convergence theorem. However, like other heuristic optimization methods, convergence here means convergence of the sequence of solutions in which all particles have converged to the points of leading particles, which is called 'premature convergence'. In the search-space, those points may or may not be the optimum.

The pseudo code of the whole process is in Algorithm 1.

We want to extend the NPO to higher dimension, in  $\mathbb{R}^d$ ,  $d \geq 3$ . We set a guiding function  $f(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_d(\mathbf{x}))^T$  where  $\mathbf{x} \in \mathbb{R}^d$  and  $f_i(\mathbf{x})$  is a real valued function in  $\mathbb{R}^d$ . Now (4.1.1) becomes the multi-dimensional Newton method

$$\mathbf{x} \leftarrow \mathbf{x} - MDf|_{\mathbf{x}}^{-1} f(\mathbf{x}), \quad (4.1.4)$$

where  $M$  is a  $d$ -by- $d$  constant matrix and  $Df|_{\mathbf{x}}^{-1}$  is an inverse of the Jacobian matrix of  $f$  at  $\mathbf{x}$ . The eigenvalues of  $M$  are related with local search tendency,  $m$ . As the eigenvalues increases from 0, the searching paths around the leading particles become more and more irregular as we

---

**Algorithm 1** Newton Particle Optimization (in  $\mathbb{C}$ )[74]

---

```

for each particle do
    Initialize particle  $z_i$  with  $m_i$ .
end for
while maximum iterations or minimum error criteria is not attained do
    for each particle do
        Calculate fitness value  $g$ .
    end for
    Choose  $n$  best members  $\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n$  in the search domain.
    Set the guiding function  $f(z) = (z - \mathbf{p}_1)(z - \mathbf{p}_2) \cdots (z - \mathbf{p}_n)$ .
    for each particle do
         $z_i \leftarrow z_i - m_i \frac{f(z_i)}{f'(z_i)}$ 
    end for
end while

```

---

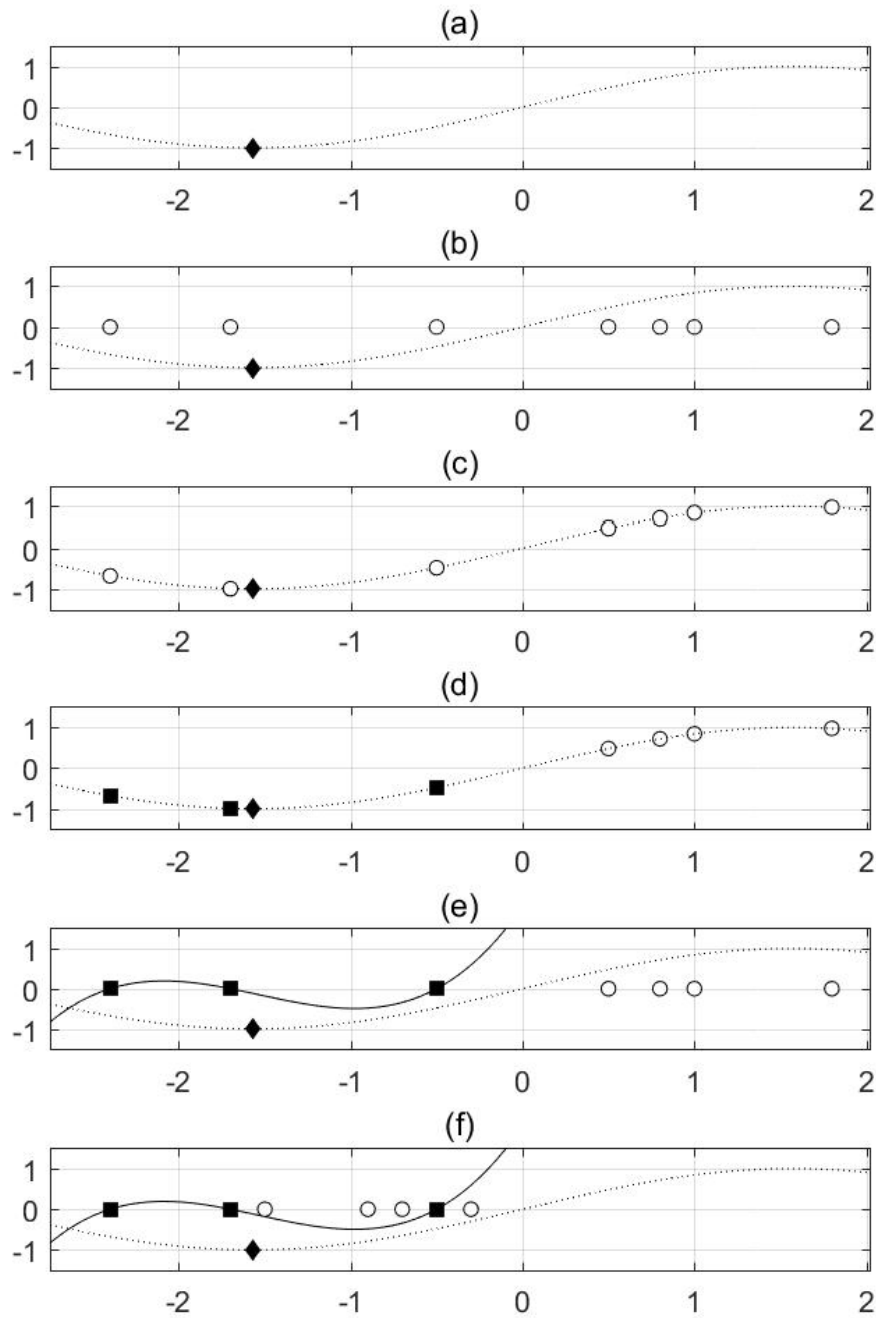
observed in 2-dimensional case in Figure 3-2 to 3-4. We will check the appropriate value of  $m$  later in this chapter.

## 4.2 Searching manner of NPO

The searching manner of NPO is in contrast to conventional methods. We compare NPO with the two typical optimizers, gradient descent method(GDM) and PSO. When it comes to using population, NPO and PSO belong to same class and GDM belong to the other class. Before we move on to this topic, it needs to be mentioned that NPO and PSO are heuristic whereas GDM is algorithmic, which is the opposite concept of heuristic. Being algorithmic is following step-by-step procedure based on the information, not on the intuition or randomness. Other than that the algorithmic way requires more information than heuristic style, they have a different type of uncertainty.

As a matter of fact, GDM has a hidden uncertainty which derives from the initial guess. GDM may give us different results depending on the initial condition not as long as the function to be optimized is convex. In order to keep the uncertainty from the initial guess off, it is safe to start with multi-agent as population-based methods do. Because the most expensive part of optimization is usually the evaluation of fitness, population-based methods are computationally heavy but it is worth using if the fitness function is complicated. Though population-based methods usually start with uniformly random distributed candidate solutions, the interaction among population lessens the impact of initial guess.

Population-based methods can relieve the uncertainty from initial guess, but most of population-based methods are stochastic and bring randomness to move particles. As we mentioned above, population-based methods are adaptable to solve the intricate problem. Without randomness, it is hard to take action for current situation flexibly. So most population-based methods include the generation of random numbers.



**Figure 4-1:** NPO Algorithm. (a) A dotted line is represented the hidden fitness function and the global optimum labeled by diamond shape should be found. (b) Distribute the particles in uniformly random manner. These particles are candidate solution and marked by white circle. (c) Evaluate the fitness of the particles. (d) Choose the top 3 best fitters as leading particles. Leading particles are marked with black square label. (e) Make the guiding function with leading particles. (f) Update the other particle's positions near the best fitter by applying Newton method with guiding function. Then repeat from (c) until it satisfies the stopping criterion.



With regard to randomness, NPO and PSO are different. NPO uses the property of chaos whereas PSO generates the random numbers to search the space. The biggest difference is the degree of freedom to control the stochasticity. NPO needs the random numbers for initialization of particles' positions and their degree coefficient  $m$ . On the other hand, PSO needs random numbers at every iteration from the initialization step. This implies that NPO is less uncertain than PSO. In addition NPO can get the same result as long as the initial condition is same. As a population based method, NPO suppresses the uncertainty from every iteration but keeps the uncertainty from initial guess, like GDM.

From the structure of the algorithm, each particle in PSO compares the best current optimum with its own private best and makes a noisy crawling toward the target. The particles in NPO do not memorize their history and simply jump toward one of the best known optimums in a diverse way.

### 4.3 Convergence of metaheuristics

For convergence of metaheuristic methods, there are three types that should be considered. First is premature convergence, which means that the particles gather somewhere and stop moving before the stopping criteria is satisfied. We want to avoid this convergence because it results in getting trapped in local minima, leading to bad performance. Many metaheuristics have an issue of premature convergence. There are some researches proposed to avoid the premature convergence. Premature convergence of a global optimizer should be checked carefully.

Second is local convergence, which means that the particles converge to a local minimum. This is the basic ability to be expected to achieve for an optimizer. Third is global convergence, which means that the particles find a global minimum. This convergence is the most desired convergence but no algorithm can guarantee this convergence regardless of the characteristics of fitness functions. Generally it is hard to distinct whether the particles are prematurely convergent, locally convergent, or globally convergent. Without local convergence, the global convergence is also not guaranteed even for a convex function. Check the definition in the following.

*Definition 4.3.1.* (Particle's convergence)[57] A particle  $i$  is said to be convergent if

$$\lim_{t \rightarrow +\infty} x_{i,t} = p$$

where  $x_{i,t}$  is the position of particle  $i$  at time  $t$ , and  $p$  is the any fixed position in the search space  $\mathcal{S}$ . The optimizer is said to be 'prematurely convergent' if all particles are convergent.

Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be a measurable function to be optimized in the search space  $\mathcal{S} \subset \mathbb{R}^d$ . The



### 4.3 Convergence of metaheuristics

function  $D$  is defined as

$$D(y_t, x_{i,t}) = \begin{cases} y_t & \text{if } f(g(x_{i,t})) \geq f(y_t), \\ g(x_{i,t}) & \text{if } f(g(x_{i,t})) < f(y_t) \end{cases} \quad (4.3.1)$$

where  $g(x_{i,t})$  denotes the application of an optimizer[57].  $y_t$  denotes the current global best at time  $t$ . The following condition is a necessary condition for local and global convergence of an optimizer.

*Definition 4.3.2.* (Algorithm condition)[57] The mapping  $D : \mathcal{S} \times \mathbb{R}^d \rightarrow \mathcal{S}$  should satisfy  $f(D(x, \xi)) \leq f(x)$  and if  $\xi \in \mathcal{S}$ , then  $f(D(x, \xi)) \leq f(\xi)$ .

From the construction,  $D$  is satisfying the algorithm condition.

*Definition 4.3.3.* (Optimality region)[57] The algorithm is said to have found a solution if it is able to generate a point in the optimality region,  $R_\epsilon$ , defined as

$$R_\epsilon = \{z \in \mathcal{S} | f(z) < \psi + \epsilon\},$$

where  $\psi$  denotes the essential infimum of  $f$ .

This following is the sufficient condition for convergence to a local minimum.

*Definition 4.3.4.* (Local convergence)[57] For any  $x_t \in \mathcal{S}$  there exists a  $\gamma > 0$  and an  $0 < \eta \leq 1$  such that

$$\mu_t(\text{dist}(x_{t+1}, R_\epsilon) \leq \text{dist}(x_t, R_\epsilon) - \gamma \text{ or } x_t \in R_\epsilon) \geq \eta$$

Let  $\{x_t\}_{t=0}^\infty$  be a sequence generated by the algorithm,  $D$ . Then

$$\lim_{t \rightarrow \infty} P(x_t \in R_\epsilon) = 1.$$

This following is the sufficient condition for convergence to a global minimum.

*Definition 4.3.5.* (Global convergence)[57] For any (Borel) subset  $A$  of  $\mathcal{S}$  with  $m(A) > 0$ ,

$$\prod_{t=0}^{\infty} (1 - \mu_t(A)) = 0$$

where  $\mu_t(A)$  is the probability of  $A$  being generated by  $\mu_t$ . Let  $\{x_t\}_{t=0}^\infty$  be a sequence generated

by the algorithm, D, then

$$\lim_{t \rightarrow \infty} P(x_t \in R_\epsilon) = 1.$$

To understand how an optimizer finds the optimum effectively, it might be a good choice to study about the convergence of an optimizer. PSO keeps moving until all particles' fitness values are same. Note that it does not mean the particles gather in a point. They can be scattered in some points as long as their fitness values are same. Whereas the particles of NPO can be scattered in a few points without reference to fitness values. NPO maintains the diversity of particles well.

Though PSO algorithm cannot guarantee the local convergence, it can be guaranteed with modification of the position of the global best particle only. F. van den Bergh et al. showed premature convergence of PSO and modified PSO to achieve local convergence[57]. This modified PSO is referred to as guaranteed convergence PSO(GCPSO). For GCPSO, the update of global best particle  $x_\tau$ , whose index is  $\tau$ , is

$$x_{\tau,t+1} = y_t + wv_{\tau,t} + \rho_t(1 - 2r_t),$$

where  $r_t$  is randomly chosen from uniformly distributed  $[0, 1]$ . The previous two terms in the update equation is same with standard PSO algorithm. Outstanding modified part,  $\rho_t$ , is defined as follows.

$$\rho_{t+1} = \begin{cases} 2\rho_t & \text{if } \#\text{successes} > s_c \\ 0.5\rho_t & \text{if } \#\text{failures} > f_c \text{ and } \rho_t > \epsilon_m \\ \rho_t & \text{otherwise} \end{cases} \quad (4.3.2)$$

where  $\epsilon_m$  represents the smallest allowable value of  $\rho$ , maybe the machine precision. A 'failure' occurs when  $f(\hat{y}_t) \geq f(\hat{y}_{t-1})$  and a 'success' occurs when  $f(\hat{y}_t) < f(\hat{y}_{t-1})$ . The number of successes, or failures is reset to zero when the failure, or success happens, respectively.

## 4.4 Local convergence of NPO

Standard NPO is prematurely convergent. A guiding function has a neighborhood convergent to the roots, called radius of convergence, by Newton method. On the other hand, finding minimum within finite candidate solution is impossible for black box problems. Thus the particles stagnate before they find the minimum.

But it can be guaranteed with variation similar to the procedure of GCPSO in the previous section[57]. We call modified NPO as guaranteed convergence NPO(GCNPO). For NPO, there needs to keep updating the positions of leading particles to avoid premature convergence. This following theorem and its proof are similar to those of PSO[57].

**Theorem 4.4.1.** (*Local convergence of GCNPO*) GCNPO is locally convergent if the particle updated with

$$x_{\tau,t+1} = y_t + \rho_t(1 - 2r_t)$$

belongs to leading particles as well as the global best particle  $y_t$  does so.  $\rho_t$  is defined in (4.3.2) and  $r_t$  is a random number in  $[0, 1]$ . By entry of random numbers, the particles keep moving in any situation and search the space densely or sparsely.

*Proof.* Randomness,  $\rho_t$  part is exactly from the GCP SO[57]. This update is about sampling a point from a hypercube whose length is  $2\rho$  centered at temporal best,  $y_t$ . Since  $\rho_t$  is changing, we denote  $\rho$  instead.  $M_k$  denotes that hypercube and  $\mu_k$  denotes the uniform probability measure defined on  $M_k$ . Premature convergence never happens because it keeps updating the position of leading particles by randomness regardless of any interaction of particles or environment. We define a compact set

$$L_0 = \{x \in \mathcal{S} : f(x) \leq f(x_0)\} \quad (4.4.1)$$

where  $x_0$  is the particle who has the largest fitness value, defined as

$$x_0 = \operatorname{argmax}_{x_i} \{f(x_i)\}, i \in \mathbb{Z}, 1 \leq i \leq N \quad (4.4.2)$$

where  $N$  is the population. Then  $y_t \in L_0$ . And  $y_t \in M_k$ . Thus  $m[M_k \cap L_0] > 0$ .  $m$  denotes the Lebesgue measure of a set. A non-degenerate sampling volume  $\mu_k$  with support  $M_k$  exists. Now we check the local convergence of GCNPO.

$\mathcal{S}$  is compact and has a non-empty interior. Then so does  $L_0$ . By definition,  $R_\epsilon \subset L_0$ . Because a closed subset of compact set is compact,  $R_\epsilon$  is compact with a non-empty interior. Refer [57] in detail. We choose a ball,  $B'$ , centered at  $c'$  in  $R_\epsilon$ .  $x'$  is the argument of  $x \in L_0$  for the maximum distance with  $c'$ .  $B$  is the hypercube centered at  $c'$  whose side length is  $2(\operatorname{dist}(c', x') - 0.5\rho)$ .  $C$  is the convex hull of  $x'$  and  $B'$ . The tangent line connecting  $B'$  and  $x'$  is the longest line of  $x'$  and  $B'$ . This implies that the volume surrounded by any other convex hull made by any  $x \in L_0$  is greater than the volume of  $C \cap B$ . Therefore for any  $x \in L_0$ ,

$$\mu_k[\operatorname{dist}(D(\hat{y}, x_\tau), R_\epsilon) < \operatorname{dist}(x, R_\epsilon) - 0.5\rho] \geq \eta = \mu[C \cap B] > 0$$

where  $\mu_k$  is the uniform distribution measure on the hypercube whose center is at  $x$  with side length  $2\rho$ . Therefore GCNPO is locally convergent.

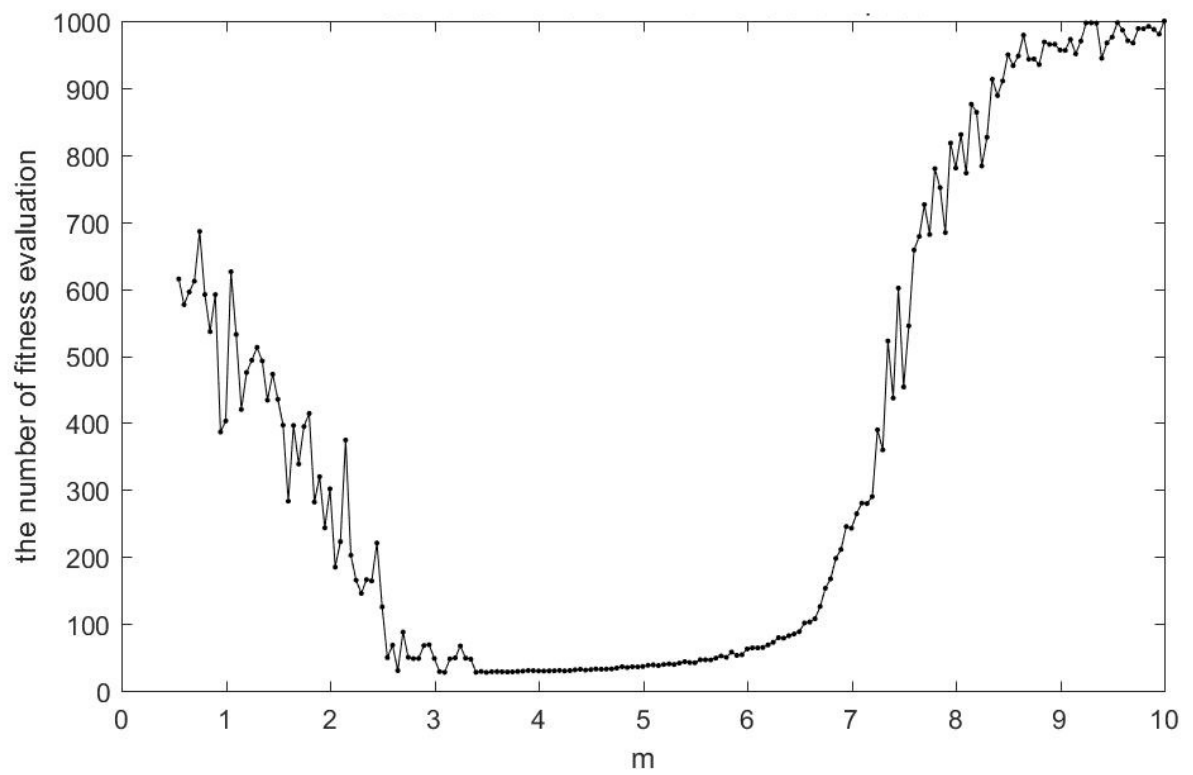
## 4.5 Criteria for choice of $m$ and $M$

We can control the trade-off between exploration and exploitation in NPO by adjusting the parameter  $m$  in (4.1.1), which is a ‘degree coefficient’. A particle’s movement is determined by its initial position and its degree coefficient  $m$ . This  $m$  affects to the movement of a particle by distance or convergence to the leading particles. The assignment of various  $m$  to population makes them follow diverse Newton path.

when it comes to modified Newton method, the sequences are bounded if their  $m$  values satisfy  $|m - n| < n$ , where  $n$  is the degree of a polynomial guiding function. The sequences are convergent if their  $m$  values satisfy  $|m - 1| < 1$ . If  $|m|$  is small, then movement of particles is also small as in Figure 3-6. Meanwhile, the particles move erratically if  $|m|$  is close to  $2n$ .

We suggest the practical region for  $m$  by experiments in Figure 4-2. Every particle is assigned to different values  $m \in \mathbb{R}^d$ . Subscript  $i$  is the dimension for  $m_i$ . We can assign same values in every dimension or not. We choose the random numbers from the interval  $[0, m_{\max}]$  uniformly for  $m_i$ . The optimal values for  $m_{\max}$  is illustrated in Figure 4-2. They show it is good to choose  $m$  around the middle of the range, that is,  $m_{\max} \approx n$ . y-axis denotes average of the required number of fitness evaluation to lower the error below the tolerance,  $10^{-8}$  for 51 multiple run. With less number of particles,  $m_{\max} > n$  is a proper choice to generate more diverse Newton paths.

For diverse behaviors of particles, we can change and mix the combination of the particles and their degree coefficient  $m$  without control of the value  $m$  itself. Standard NPO assigns fixed  $m$  for every particle. But the particles can be aligned in the order of their fitness values. This process is deterministic but hard to be expected so that it is like an uniformly mixing effect. From initial choice of  $m$ , we have well mixed random numbers,  $m$  and match the particles and  $m$  in fitness ranking order. In standard NPO, if a particle has a big  $|m|$ , this particle is working for the exploration all the time. But if we assign  $m$  for fitness ranking this gives the particles flexibility.



**Figure 4-2:** Performance according to  $m_{max}$ . The parameter settings are 200 particles with 4 leading particles for rosenbrock function.[74]

## 5

# Construction of A Guiding Function

We present the conditions for an ideal guiding function for Newton Particle Optimizer in this chapter[74]. Generating a proper guiding function is the main issue to balance between the exploration and the exploitation. In the previous chapter, we naturally came up with the guiding function in 2-dimensional space using complex system. However, in higher dimension, it is problematic to make an adaptable guiding function. We check the qualification of a guiding function and propose how to construct a guiding function in multi-dimensional space.

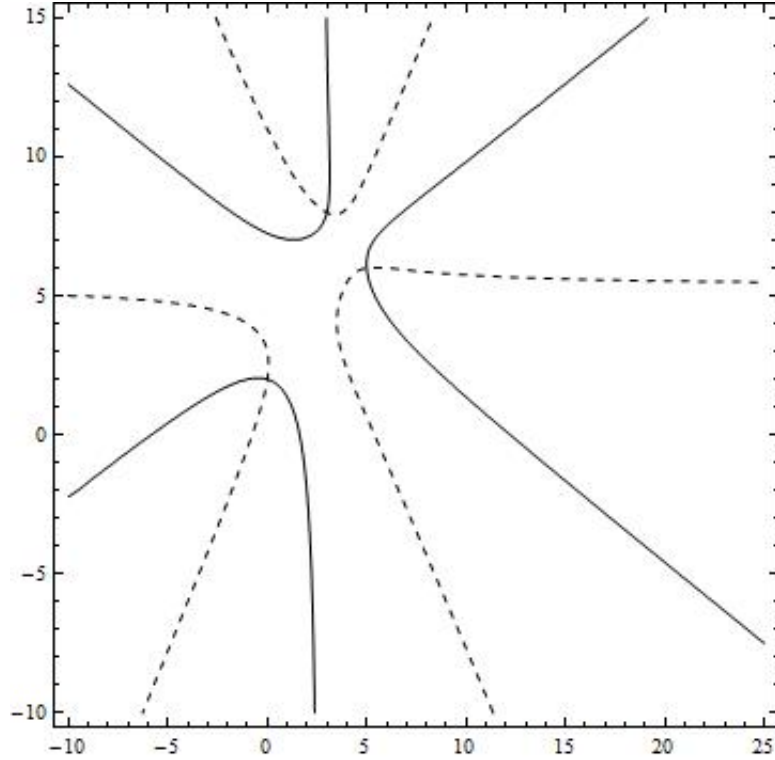
### 5.1 Conditions for an ideal guiding function

Taking a proper guiding function is closely related to the performance of NPO. There are three conditions for an ideal guiding function to be satisfied as follows[74].

- (1) the function has zeros at the designated points and no zeros elsewhere.
- (2) the function is symmetric.
- (3) the inverse of its Jacobian is easy to compute.

The first condition is essential for the guiding function to attract particles to the roots. The second condition is for uniform search without bias. The third condition is for the efficiency of computation from Newton method. This is relatively easy to satisfy as long as the guiding function is a polynomial function. The proposed factored polynomial as (4.1.2) in 1 and 2-dimensional search space satisfies all those conditions. But unfortunately, there are no such factored polynomials in more higher dimensional space. Thus we give up the first condition partly, to construct a guiding function in  $\mathbb{R}^d$ ,  $d \geq 3$ .

Still, this guiding function is satisfying with having zeros at the designated points but it may have other zeros in the search space. We call these undesired zeros as ‘phantom’. As a matter of fact, the phantoms help particles to explore the space. The Newton method drives



**Figure 5-1:** A nullcline of a factored guiding function in 2-dimensional space as (4.1.2).

other particles to the point where the function has zeros, whether it is a desired point or not. Phantoms slow down the convergence preventing the particles from gathering only near the leading particles. This is an undesired phenomenon especially in low dimension because the search space is so small that NPO can find the good enough candidate optimum soon. But for higher dimensional search space, existence of phantoms is not an issue. Leading particles are changed depending on their fitness values at every iteration step, then phantoms are changed if one of the leading particle is changed. Though we have some phantoms at every step, it will be gone easily whereas best fitters survive as long as it fits well to the problem.

## 5.2 Extension of a guiding function

We suggest a natural extension of (4.1.2) in a higher dimensional search space. (4.1.2) can be rewritten in the complex form as  $f(x + iy) = u(x, y) + iv(x, y)$  and  $\mathbf{p}_j = q_j + ir_j$ . When the number of leading particles are 3, the corresponding component functions are

$$\begin{aligned} u(x, y) &= \prod_{k=1}^n (x - q_k) - \sum_{l=1}^n (x - q_l) \prod_{m=1, m \neq l}^n (y - r_m), \\ v(x, y) &= - \prod_{k=1}^n (y - r_k) + \sum_{l=1}^n (y - r_l) \prod_{m=1, m \neq l}^n (x - q_m) \end{aligned} \quad (5.2.1)$$

## 5.2 Extension of a guiding function

where  $n$  is the number of leading particles. We show the nullcline of this guiding function in Figure 5-1. There are three intersections which would be the positions of leading particles.

When the number of leading particles are 4, the corresponding component functions are

$$\begin{aligned} u(x, y) &= \prod_{k=1}^n (x - q_k) - \sum_{m=1}^n \sum_{l=1, l>m}^n (x - q_m)(x - q_l) \prod_{k=1, k \neq l, m}^n (y - r_k) + \prod_{k=1}^n (y - r_k), \\ v(x, y) &= - \sum_{l=1}^n (x - q_l) \prod_{m=1, m \neq l}^n (y - r_m) + \sum_{l=1}^n (y - r_l) \prod_{m=1, m \neq l}^n (x - q_m). \end{aligned} \quad (5.2.2)$$

As we see, the form is changed whether the number of leading particles is an odd or even number regularly. Now we propose, as an extension of (5.2.1), a general guiding function in  $\mathbb{R}^d$  as

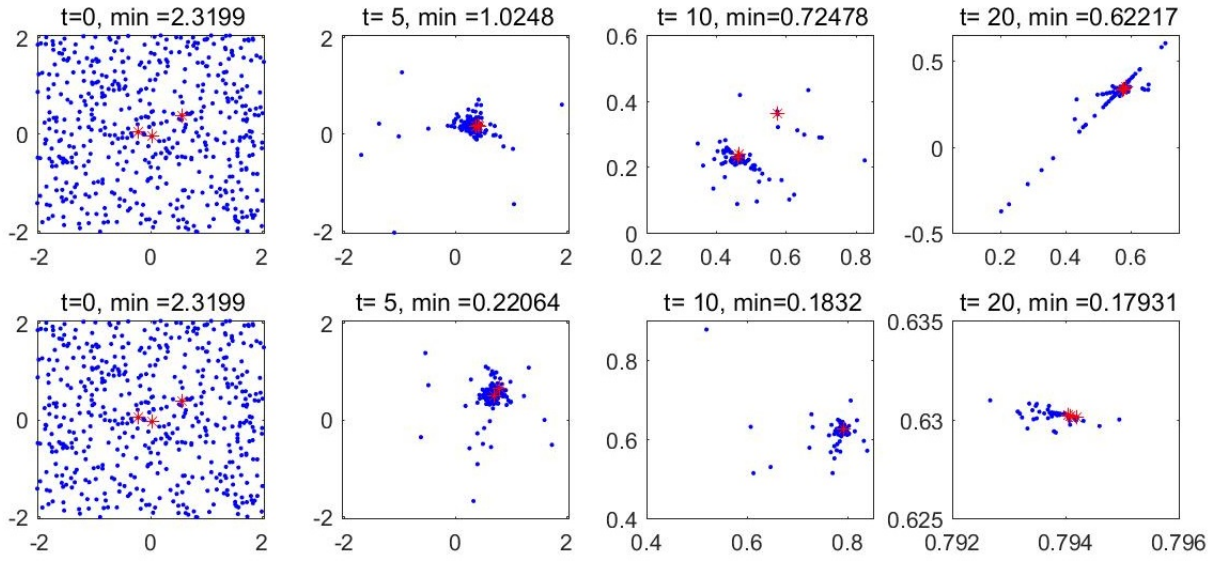
$$f_j(\mathbf{x}) = (-1)^{j+1} \left\{ \prod_{k=1}^n (x_{j+2} - p_{j+2,k}) - \sum_{l=1}^n (x_j - p_{j,l}) \prod_{m=1, m \neq l}^n (x_{j+1} - p_{j+1,m}) \right\} \quad (5.2.3)$$

where  $\mathbf{p}_i = (p_{i,1}, \dots, p_{i,d})^T$ ,  $1 \leq i \leq n$  denotes the position of a leading particle. Here we use a circular indexes like  $x_{d+1} = x_1$  in (5.2.3).

Though (5.2.3) is an extension of the case when the number of leading particles is 3, it can be used for more than 3 particles. The guiding function  $f(\mathbf{x})$  may have zero vectors at other than  $\mathbf{p}_1, \dots, \mathbf{p}_n$  in  $\mathbb{R}^d$ ,  $d \geq 3$  as long as  $f(\mathbf{x})$  vanishes if  $\mathbf{x} = \mathbf{p}_i$ ,  $1 \leq i \leq n$ . As we mentioned in the previous section, such extra zeros do not lower the searching performance much. Such phantom roots appear irregularly and helps the particles to search the space globally.

We attach some figures about the proposed guiding function in Figure 5-2. These figures represent the first two components of the position of all particles in 3-dimensional search space. As iteration goes on, the particles are finding the minimum of rosenbrock function. The iteration step and current minimum is denoted in the title. The leading particles are marked with asterisk. We compare the proposed guiding function and wrong guiding function, which does not satisfy the first two proposed conditions. In the above row in Figure 5-2, the particles gather near the leading particles from every direction for the first few step, but it finally follows a line as in the pictures at  $t = 20$ . In the below row, whereas the particles gather near the leading particles from every direction for proposed guiding function. This is desired because we want the particles to search the space all around uniformly. Though the initial position and  $m$  are all same in Figure 5-2, the performance of a proposed guiding function is better than that of a biased guiding function, as denoted in their titles. The current best values are 0.62217, 0.17931 at  $t = 20$  for both guiding functions, respectively. The proposed guiding function is reasonable.





**Figure 5-2:** The distribution of particles using a biased guiding function(above) and a proposed guiding function(below), with same initial condition for the position of particles and degree coefficient  $m$ .  $t$  counters the iteration step for NPO.  $min$  denotes temporal best.

### 5.3 Criteria for choice of leading particles

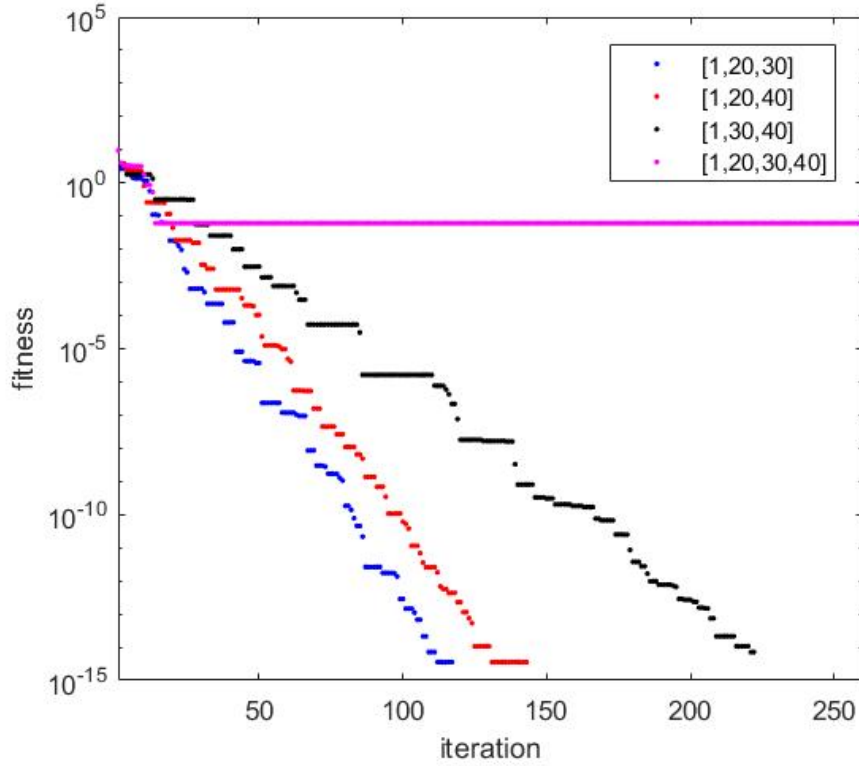
We can control the balance between exploration and exploitation with the choice of leading particles  $\mathbf{p}_1, \dots, \mathbf{p}_n$ . Their relative positions affects other particles' movement because they are ingredients of a guiding function. We suggest the choice of the temporal best fitters in (4.1.3). But this may result in lack of diversity and premature convergence to current optimums, preventing the particles from keeping searching the better one.

This is about parameter setting for implementation. The number of leading particles,  $n$  is a hyper parameter, which is usually determined in  $3 \leq n \in \mathbb{N} \leq 5$  by trial and error, for the population size of  $100 \sim 1000$  particles. Determination of  $n$  affects to the performance as in Figure 5-3.

#### 5.3.1 Idle leaders in leading particles

To prevent the premature convergence, or getting trapped at local minima, we need some idle leaders. Idle leaders mean mediocre fitters who are not the best fitters. By experiment, in Figure 5-4, we compare two cases of leading particles, one of which uses top 5 best fitters(above) and the other uses top 4 best and 1 mediocre fitter with a low rank(below). The below case with idle leader is worse then the above at first, it finds the better temporal best in the end successfully. Again, the choice of leading particles has something to do with a balance between exploitation and exploration. The idle leaders seem useless but actually they stop the particles from focusing on a small region. Once the particles gather in a small region, they lose the power to search the space globally in many metaheuristics because the particles' momentum comes

### 5.3 Criteria for choice of leading particles



**Figure 5-3:** The number of leading particles is important. Label denotes the ranking of leading particles among 100 particles. It shows the results of  $n = 3$  is better than that of  $n = 4$ .

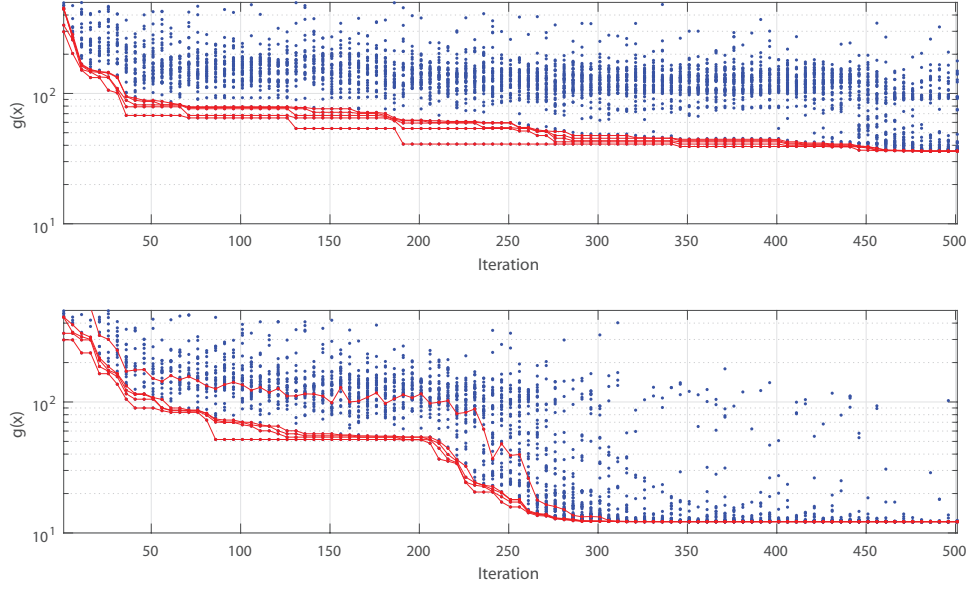
from their interaction. Diversity is essential for global search. Organization of best fitters as leading particles dominate all other particles easily. This seems efficient in the short term but it does not in the long term. Though it takes time to find a good candidate for idle leaders, it is worth of taking up the challenge to explore the unknown space. Otherwise we face the limit that those best fitters set up potentially.

#### 5.3.2 Personal best in leading particles

In comparison with PSO, NPO differs in many respects. PSO moves particles referring the personal best and global best whereas NPO moves mostly considering the temporal global best, which are the leading particles. To contemplate the cognitive impact from personal best like PSO, we can introduce the personal best in leading particles for NPO. We organize leading particles as two best fitters, one mediocre fitter, and one personal best,  $p_t^j$  of a fitter  $j$ . The personal best at  $t$  is

$$p_t^i = \begin{cases} x_t^i & \text{if } f(x_t^i) < f(p_{t-1}^i) \\ p_{t-1}^i & \text{otherwise} \end{cases}$$

## 5.4 Particles outside the boundary



**Figure 5-4:** Choice of leading particles affects the result: the above adopts top 5 rank performers out of 100 particles as leading particles. The below uses four best fitters and one 40th rank fitter. The blue and red dots indicate the cost values of ordinary and leading particles, respectively.[74]

where  $x_t^i$  is the position vector of the  $i$ -th particle at iteration  $t$  step, as in PSO.

Indeed, it would be profitable to stay leading particles near every local minima so that other particles improve the leading particles by exploitation. Introducing the personal best of a particle plays that kind of role in NPO. To take best advantage of the personal best in leading particles near the local minimum, the degree coefficient  $m$  of those particles attracting the leading particle should lie in the bounded region, not in a convergence region. If the personal best is not the temporal best in leading particles, then it is expected that it may slow down the global convergence but it gives a chance to improve the personal best and find local minimum. This introduction of the personal best is about strong exploitation, not about the exploration.

## 5.4 Particles outside the boundary

We usually solve the optimization problem in the bounded domain. It means that the particles for valid candidate solution should stay in the bounded region. But what if the particles get out the boundary?

A standard way is confining a particle if it gets out of boundary. A particle would be on the boundary on compulsion if it goes out. Though this way is simple but the optimizer wastes the computational source too much just for the boundary. Or we can use modulo operations, which we move the particle on the position of left-over part in division of length of bounded domain. This way gives an effect of randomness and makes the system more complicated and unexpected.

Another way is staying the position if the position of a particle turned out to be outside of boundary at the next step, potentially[18]. Or we can control the velocity vector. The velocity vector is the update vector of a position. Many methods like PSO, FO, etc, confined the velocity vector to  $V_{max}$ . This seems just for staying the particles in the domain, but it has something to do with the convergence issue[6]. These all methods to control the particles outside boundary are good but NPO considers them in a different way.

NPO leaves them wander around even though the particles are out of boundary because they will come back as long as the leading particles are in the boundary. The thing we have to do is preventing the leading particles in the boundary. It is simply settled because we give penalty to the particles outside the boundary not to be chosen as leading particles. We do not need to calculate the fitness of particles outside and need to just put the big numbers in their fitness values. This way is easy and not artificial, and has no randomness. As we mention in §4.4, the position would be bounded if  $|m - n| < n$  for a guiding function whose degree of a polynomial is  $n$ .

## 5.5 Exploration indicator

Since the balance between exploration and exploitation is so important, we suggest a measure to indicate how much the parameter is set for the exploration[74]. We define an *exploration indicator*(EI) as

$$EI_1 = \frac{m_{\max}}{2n}. \quad (5.5.1)$$

Note  $0 < EI_1 < 1$ . If  $EI_1$  is small, the particles tend to exploit more than explore the search space. As  $EI_1$  is nearly 1, they widely explore the search space. Usually the parameter is set as  $EI_1 \approx 0.5$ .

In NPO, we can control the trade-off with not only the degree coefficient  $m$  but also the choice of leading particles. Thus we suggest another exploration indicator. We define

$$EI_2 = \frac{1}{2} - \frac{n(n+1)}{4\sigma} + \frac{n(n+1)}{4(nN + n - \sigma)} \quad (5.5.2)$$

where  $\sigma$  is the summation of the ranks of the leading particles. Note  $0 < EI_2 < 1$ . If  $EI_2$  gets closer to 0, this implies that the leading particles are chosen from temporal best fitters, say, low exploration(high exploitation). Contrastively, if  $EI_2$  gets closer to 1, this implies that all leading particles come from mediocre fitters, implying high exploration(low exploitation). In Figure 5-4, the exploration indicators are (a)  $EI_2 = 0.015$  and (b)  $EI_2 = 0.366$ , respectively.

## 6

# Results

We show the experimental results of Newton particle optimizer(NPO) for various global optimization problems and compare the performance with those of particle swarm optimizer(PSO) and firefly optimizer(FO) in this chapter.

Considering a generality of black-box problems, it is hard to evaluate how good a global optimizer is. By the no-free-lunch theorem[62], there are no such algorithms that are always dominant for all problems. So evaluation for some optimizers may seem meaningless but there still exist the better optimizers for the specific problems. Here we adopt some popular test functions in 2-dimensional space and CEC 2013 problem set, which is widely used for evaluation of optimizers for higher-dimensional problems.

Refer that metaheuristic methods are interactive systems. They should be adjusted themselves in any circumstance. A hyper-parameter is a parameter set before the iterations and usually set by human by trial and error. Least human intervention is desired for automation, and a fixed parameter is not adaptable for changing environment. Thus the less the number of hyper-parameters is, the better the optimizer is. What is more, the performances of many optimizers are significantly affected by the parameters. Some parameters controls the balance between exploration and exploitation. Other parameters determine the position somewhere between the private best and global best. These parameters are artificial indeed. It is good to know the adaptable parameters before the implementation in advance, which is almost impossible. The best thing is removing the hyper-parameters and the second best thing is making the control of parameters easy like NPO. We show some results and the results are changeable under the different parameter set. But from the perspective of finding adaptable parameters, NPO is more beneficial than PSO, and FO.

## 6.1 Tests in 2-dimensional search space

First, we show the simple results in 2-dimensional space.

### 6.1.1 2-dimensional test functions

We introduce 16 test functions for fitness functions and classify those functions into 3 classes depending on the number of local minima[42]. It is no wonder that existence of many local minima makes the problem harder. In Table 6-1, functions from no.1 to no.4 have a unique global minimum and no local minima. Functions from no.5 to no.12 may have not global minimum and have several local minima. The others have many local minima. The number of local minima affects the performance significantly because getting trapped in a local minimum prevent the particles from searching a global minimum. Visualization of the functions is attached in Figure 6-1,6-2,6-3.

### 6.1.2 NPO v.s. PSO in 2D

We display the results in Figure 6-4, obtained with functions in Table 6-1. Figure 6-4 shows the performances of NPO and PSO, respectively. The first figure (a) is the median values of 51 multiple runs for the minimization of the functions. It shows almost similar performances for NPO and PSO in (a). The second figure (b) is the mean values of 51 multiple runs of the number of iterations that required to get the minimum. If the minimum found by an optimizer is attained to the tolerance  $10^{-10}$ , then the optimizer stops iteration. Maximum iteration number is 1000. It is notable that NPO converges to the global minimum so fast except some fitness functions, no. 7,8,14. On the other hand PSO converges much slower than NPO in (b), but its probability to attain the minimum is 1 as we see in (c). This happens because of the hyper-parameters. The parameters in NPO focus more on strong exploitation and those in PSO more on strong exploration. In (d), we can check the running time.

The detailed parameters are follows. Population size  $n$  is 100 for both methods and the initial position of particles are same to remove the advantage from initial position. For NPO, the matrix  $M$  for each particle was chosen such that its eigenvalues lie between 0.5 and  $\sqrt{8} \approx 2.83$ . We use 3 leading particles from top 3 best fitters for strong exploitation. The parameters for PSO were set to the recommended values which are widely used in the benchmark tests[5, 21, 30].

## 6.2 Tests in the high dimensional search space

**Table 6-1:** Test Functions[42]. Each minimum of the functions is 0.

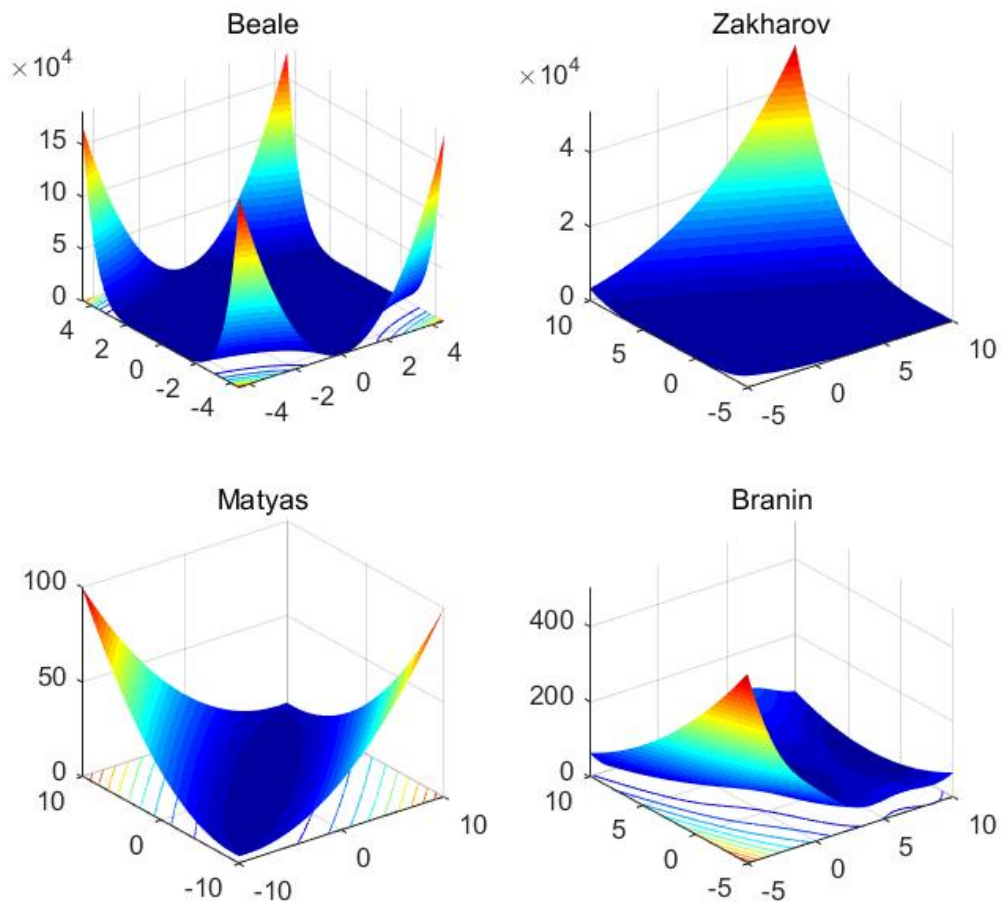
No.	Function	Domain
1	Beal $(1.5 - x_1(1 - x_2))^2 + (2.25 - x_1(1 - x_2^2))^2 + (2.625 - x_1(1 - x_2^3))^2$	$[-4.5, 4.5] \times [-4.5, 4.5]$
2	Zakharov $x_1^2 + x_2^2 + (0.5x_1 + x_2)^2 + (0.5x_1 + x_2)^4$	$[-5, 10] \times [-5, 10]$
3	Matyas $0.26(x_1^2 + x_2^2) - 0.48x_1x_2$	$[-10, 10] \times [-10, 10]$
4	Branin $-0.3978873 + (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10$	$[-5, 10] \times [0, 15]$
5	Goldstein Price $-3 + (1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)) \times (30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2))$	$[-2, 2] \times [-2, 2]$
6	Easom $1 - \cos(x_1)\cos(x_2)$ $\exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$	$[-2\pi, 2\pi] \times [-2\pi, 2\pi]$
7	Drop-wave $1 - \frac{1 + \cos(12\sqrt{x_1^2 + x_2^2})}{0.5(x_1^2 + x_2^2) + 2}$	$[-5.12, 5.12] \times [-5.12, 5.12]$
8	Rosenbrock $100(x_2 - x_1^2)^2 + (x_1 - 1)^2$	$[-5, 10] \times [-5, 10]$
9	Ackley $e - 20 \exp(-0.2\sqrt{\frac{1}{2}(x_1^2 + x_2^2)}) - \exp(\frac{1}{2}(\cos(2\pi x_1) + \cos(2\pi x_2))) + 20$	$[-15, 30] \times [-15, 30]$
10	Perm $\sum_{i=1}^2 \left( \sum_{j=1}^2 (j^i + 0.5) \left( (\frac{x_j}{j})^i - 1 \right) \right)^2$	$[-2, 2] \times [-2, 2]$
11	Six-hump camel back $1.0316284 + (4 - 2.1x_1^2 + \frac{x_1^4}{3})x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$	$[-3, 3] \times [-2, 2]$
12	Michalewicz $1.8013034 - \sin(x_1)\sin^{20}(\frac{x_1^2}{\pi}) - \sin(x_2)\sin^{20}(\frac{2x_2^2}{\pi})$	$[0, \pi] \times [0, \pi]$
13	Shubert $\left( \sum_{i=1}^5 i \cos((i+1)x_1 + i) \right) \left( \sum_{i=1}^5 i \cos((i+1)x_2 + i) \right) + 186.7309088$	$[-10, 10] \times [-10, 10]$
14	Rastrigin $20 + x_1^2 - 10 \cos(2\pi x_1) + x_2^2 - 10 \cos(2\pi x_2)$	$[-5.12, 5.12] \times [-5.12, 5.12]$
15	Schwefel $837.9657745 - x_1 \sin(\sqrt{ x_1 }) - x_2 \sin(\sqrt{ x_2 })$	$[-500, 500] \times [-500, 500]$
16	Levy $w_i = 1 + (x_i - 1)/4, i = 1, 2.$ $(w_1 - 1)^2(1 + 10 \sin(\pi w_1 + 1)^2) + (w_2 - 1)^2(1 + \sin^2(2\pi w_2)) + \sin^2(\pi w_1)$	$[-10, 10] \times [-10, 10]$

## 6.2 Tests in the high dimensional search space

This section compares numerical performances of NPO with those of PSO and FO in the high dimensional search space. We tested with 28 benchmark functions suggested from the CEC 2013 competition for real-parameter optimization[33]. CEC 2013 problem set is including with 5 unimodal functions, 15 multi-modal functions and 8 composite functions in range  $[-100, 100]^d$ , where  $d$  is the dimension of search space.



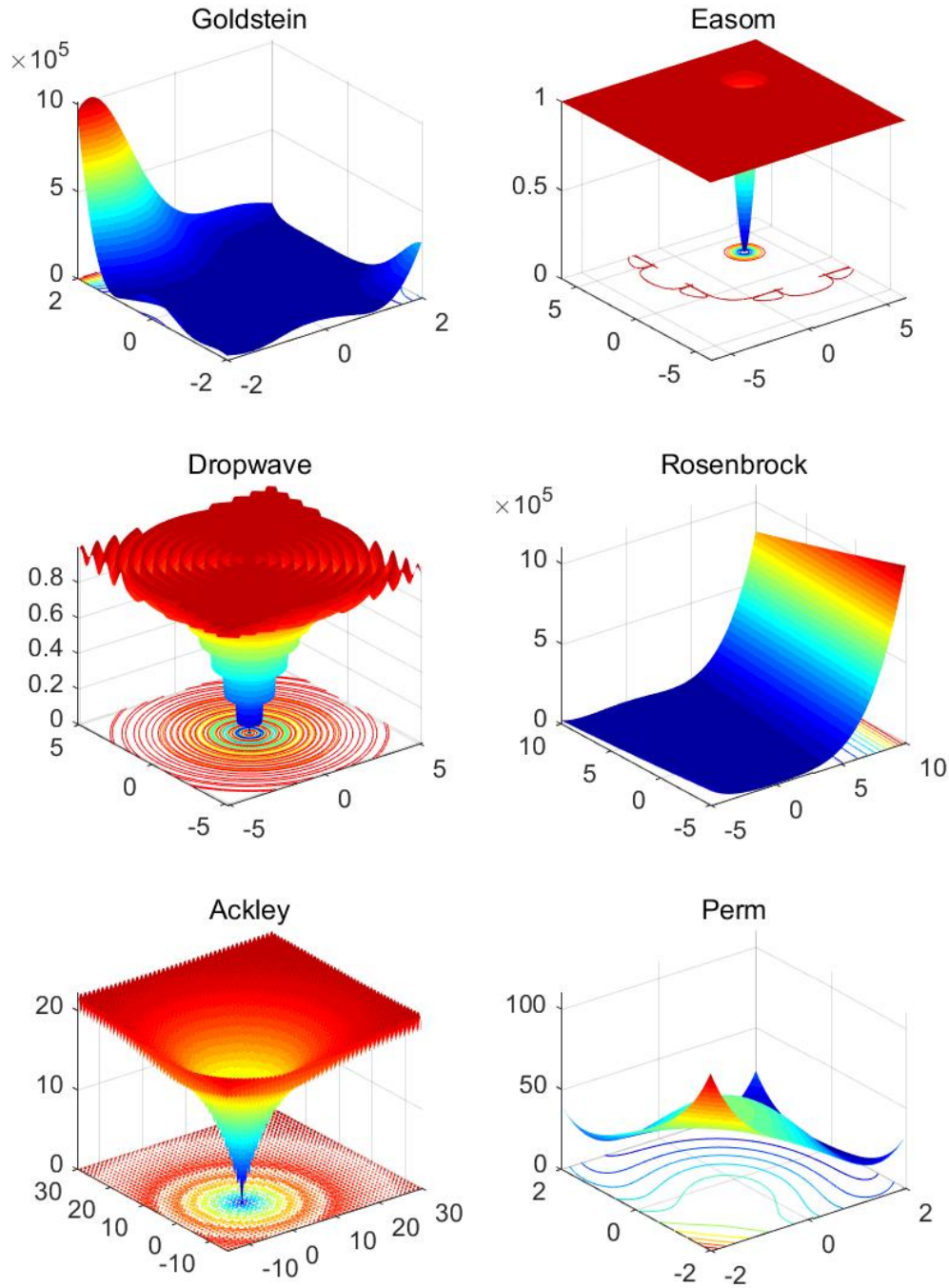
## 6.2 Tests in the high dimensional search space



**Figure 6-1:** Test functions which have no local minima.

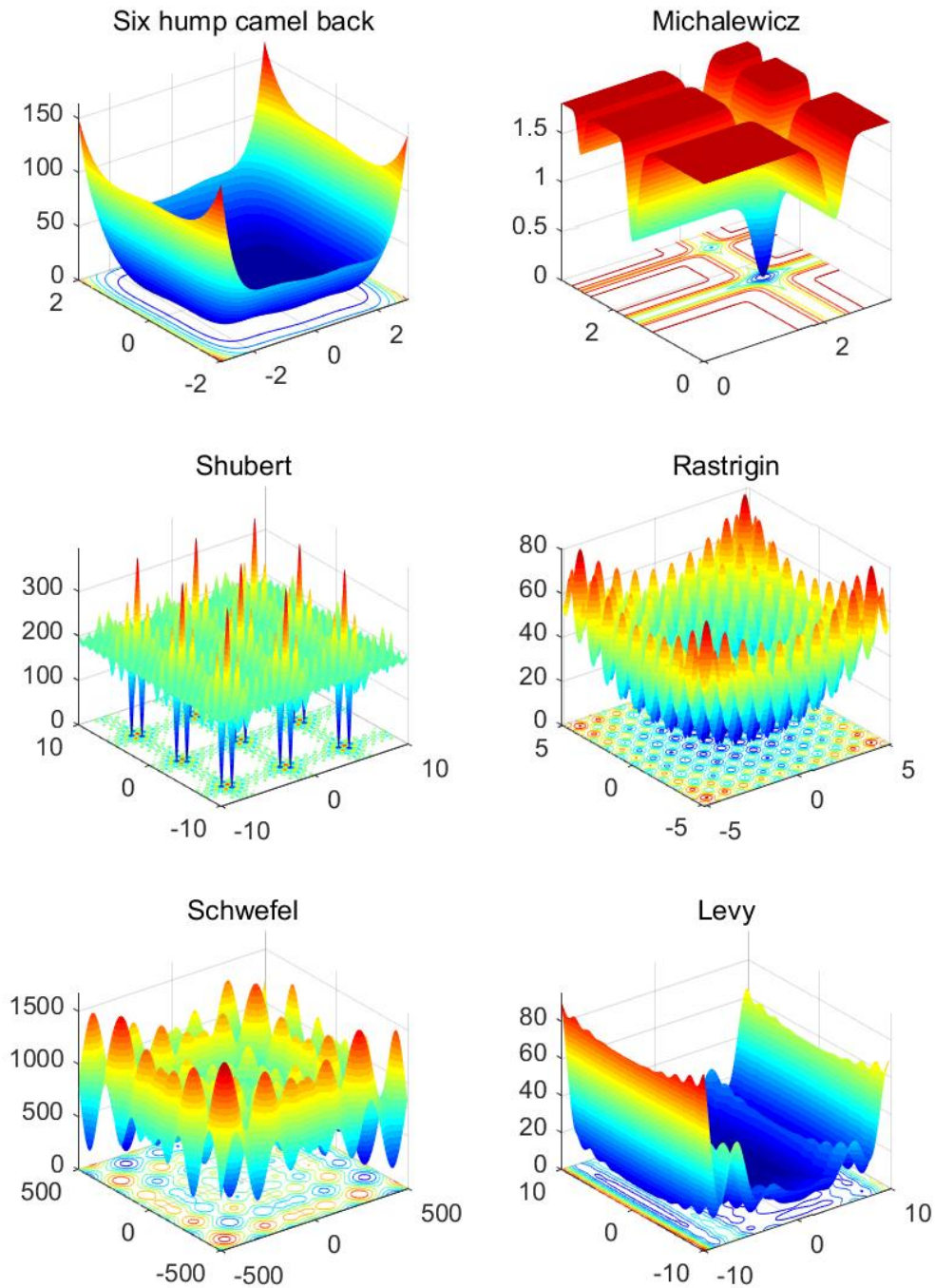


## 6.2 Tests in the high dimensional search space



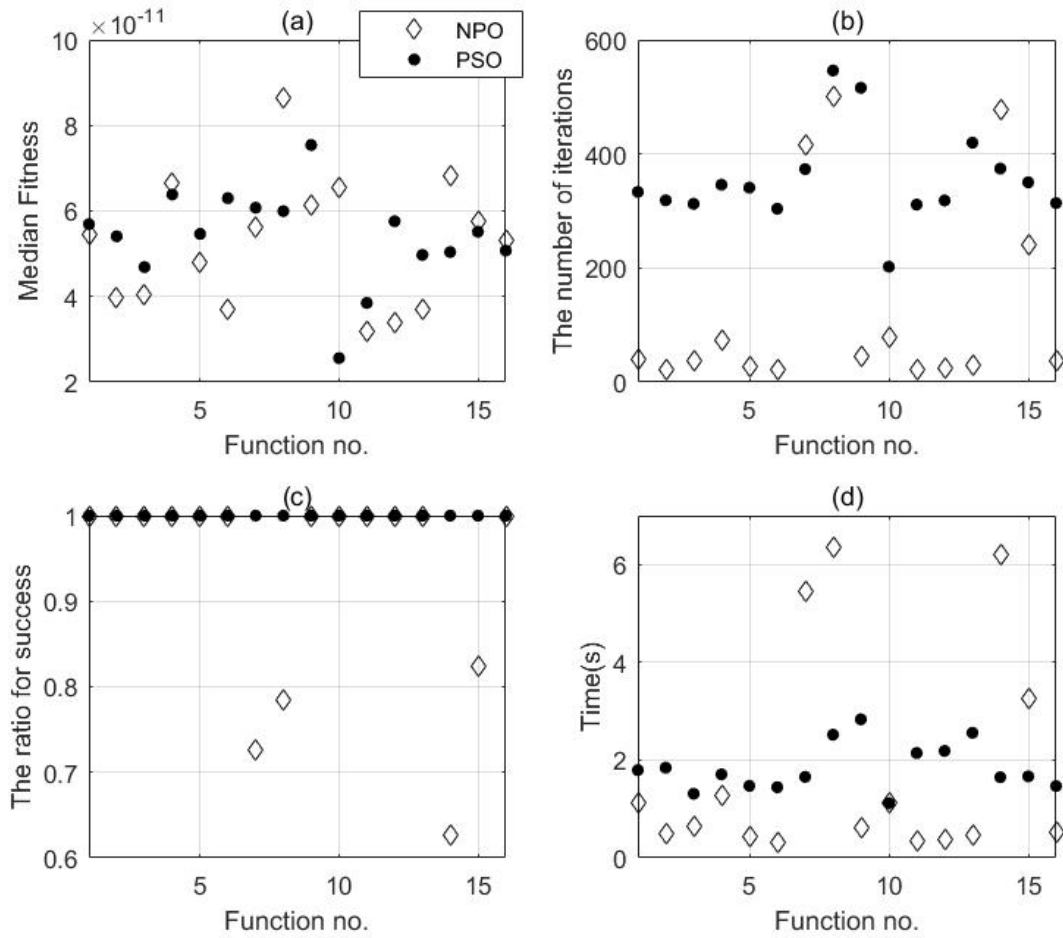
**Figure 6-2:** Test functions which have several local minima.

## 6.2 Tests in the high dimensional search space



**Figure 6-3:** Test functions which have many local minima.

## 6.2 Tests in the high dimensional search space



**Figure 6-4:** 2-dimensional results of test function in Table 6-1.

### 6.2.1 10-dimensional search space

The fitness functions lie in 10-dimensional space and 51 multiple runs are implemented for each test function. We limit the number of evaluations of functions as dimension times 10000, that is, 100000 for 10-dimensional problems.

The detailed parameters are followed. NPO uses 4 leading particles, 3 from best fitters and one from the 50% performer. The matrix  $M$  for each particle was chosen such that its eigenvalues lie between 0 and 4. The parameters for PSO were set to the recommended values which are widely used in the benchmark tests[5, 21, 30]. The parameters for FO were taken from [56, 41].

Table 6-2 shows that performances of three schemes are comparable. However, the summarized mean ranking of NPO, PSO and FO in Table 6-3 are respectively 1.821, 2.107 and 2.01, which indicates that NPO is practically better than PSO and FO with these benchmark functions.

### 6.2.2 30-dimensional search space

As the dimension becomes higher, NPO and PSO put a new complexion on the performance. NPO tends to give better result than PSO in higher dimension. Because NPO can drive the particles to temporal best with strong exploitation. The suggested number of fitness evaluation is the multiplication of dimension and 10000. It is merely increased linearly while the search space gets larger exponentially. Therefore under CEC 2013 problems' suggestion, NPO has a preponderance over PSO and FO.

On the other hand, for the computational cost, NPO is more expensive than PSO. Because NPO includes inverse computations for matrix inside at every iteration. Whereas the computational cost of PSO increases linearly as the dimension is higher. Thus we have to choose the proper metaheuristic methods in a given environment for trading off between computational cost and accuracy.

The detailed parameters are as follows. NPO uses 3 leading particles, all from best fitters. The matrix  $M$  for each particle was chosen such that its eigenvalues lie between 0 and 3. The parameters for PSO were set to the recommended values which are widely used in the benchmark tests[5, 21, 30]. The parameters for FO were taken from [56, 41] except for  $\delta=0.02$ .

Table 6-4 shows that performances of three schemes are comparable. However, the summarized mean ranking of NPO, PSO and FO in Table 6-5 are respectively 1.5, 2.357 and 2.143, which indicates that NPO is practically better than PSO and FO with these benchmark functions.

## 6.2 Tests in the high dimensional search space

Function		Best	Worst	Median	Mean	Variance
1	NPO	1.000e-08	1.000e-08	1.000e-08	1.000e-08	1.000e-08
	PSO	1.000e-08	1.000e-08	1.000e-08	1.000e-08	1.000e-08
	FO	2.413e-06	1.526e-05	8.771e-06	9.311e-06	8.862e-12
2	NPO	1.935e+04	1.056e+06	2.269e+05	3.291e+05	2.670e+05
	PSO	3.686e+04	2.911e+06	2.767e+05	4.520e+05	5.253e+05
	FO	2.720e+03	2.255e+05	3.900e+05	5.487e+04	2.480e+09
3	NPO	9.035e-01	1.234e+08	9.889e+05	5.881e+06	1.765e+07
	PSO	4.078e-03	4.513e+07	1.077e+05	1.631e+06	6.624e+06
	FO	2.429e+00	1.877e+03	6.572e+01	2.622e+02	1.854e+05
4	NPO	4.565e+01	1.671e+03	2.231e+02	3.531e+02	3.360e+02
	PSO	3.972e+02	3.814e+03	1.223e+03	1.340e+03	6.479e+02
	FO	1.162e+02	5.479e+03	1.311e+03	1.655e+03	1.514e+06
5	NPO	2.130e-05	3.202e-04	1.141e-04	1.232e-04	6.380e-05
	PSO	1.000e-08	1.000e-08	1.000e-08	1.000e-08	1.000e-08
	FO	7.248e-04	2.908e-03	1.652e-03	1.709e-03	2.628e-07
6	NPO	4.910e-06	7.939e+01	9.853e+00	7.926e+00	1.107e+01
	PSO	4.753e-02	7.699e+01	1.013e+01	8.818e+00	1.053e+01
	FO	3.314e-03	9.820e+00	9.814e+00	9.440e+00	3.505e+00
7	NPO	8.153e-03	1.650e+01	1.586e+00	3.488e+00	4.058e+00
	PSO	4.003e-01	8.370e+01	2.790e+00	7.079e+00	1.384e+01
	FO	7.785e-03	1.504e-01	2.231e-02	3.281e-02	8.395e-04
8	NPO	2.008e+01	2.047e+01	2.032e+01	2.030e+01	9.520e-02
	PSO	2.009e+01	2.047e+01	2.033e+01	2.031e+01	8.881e-02
	FO	2.014e+01	2.050e+01	2.038e+01	2.036e+01	6.315e-03
9	NPO	1.324e+00	6.142e+00	3.495e+00	3.493e+00	1.098e+00
	PSO	7.067e-01	5.864e+00	3.123e+00	3.045e+00	1.214e+00
	FO	4.928e-02	2.492e+00	1.644e+00	1.500e+00	3.757e-01
10	NPO	9.353e-02	1.202e+00	4.733e-01	5.136e-01	2.461e-01
	PSO	3.693e-02	1.321e+00	3.151e-01	3.828e-01	2.482e-01
	FO	1.751e-02	2.833e-01	8.426e-02	9.841e-02	3.595e-03
11	NPO	8.060e-09	2.989e+00	4.997e-01	6.962e-01	7.574e-01
	PSO	1.000e-08	4.975e+00	1.990e+00	1.974e+00	1.272e+00
	FO	9.950e-01	1.492e+01	4.975e+00	5.736e+00	8.497e+00
12	NPO	2.984e+00	2.984e+01	1.293e+01	1.356e+01	5.561e+00
	PSO	2.985e+00	3.927e+01	1.293e+01	1.435e+01	7.071e+00
	FO	9.950e-01	1.691e+01	5.970e+00	6.438e+00	1.150e+01
13	NPO	7.426e+00	3.828e+01	2.293e+01	2.271e+01	8.047e+00
	PSO	4.003e+00	3.909e+01	2.258e+01	2.215e+01	7.815e+00
	FO	1.990e+00	2.745e+01	1.046e+01	1.129e+01	4.502e+01
14	NPO	2.995e-01	2.894e+02	7.649e+01	1.087e+02	8.823e-01
	PSO	3.747e-01	5.912e+02	1.018e+02	1.257e+02	1.114e+02
	FO	1.189e+01	9.208e+02	4.004e+02	4.424e+02	3.466e+04
15	NPO	2.676e+01	1.442e+03	8.231e+02	8.216e+02	3.104e+02
	PSO	2.846e+02	1.555e+03	7.610e+02	7.990e+02	2.823e+02
	FO	1.303e+02	8.364e+02	2.537e+02	2.984e+02	2.719e+04
16	NPO	5.424e-02	1.021e+00	4.327e-01	4.365e-01	2.474e-01
	PSO	5.023e-01	1.634e+00	9.331e-01	9.553e-01	2.372e-01
	FO	1.304e-02	3.961e-01	8.392e-02	1.140e-01	7.635e-03
17	NPO	1.097e+01	1.985e+01	1.405e+01	1.464e+01	2.113e+00
	PSO	2.308e+00	2.157e+01	1.440e+01	1.376e+01	4.639e+00
	FO	1.242e+01	3.003e+01	1.757e+01	1.808e+01	1.674e+01
18	NPO	1.400e+01	4.754e+01	2.411e+01	2.558e+01	7.811e+00
	PSO	7.196e+00	5.724e+01	3.009e+01	3.051e+01	9.823e+00
	FO	1.207e+01	2.984e+01	1.687e+01	1.770e+01	1.698e+01
19	NPO	2.148e-01	1.299e+00	5.489e-01	5.837e-01	2.068e-01
	PSO	3.495e-01	1.106e+00	6.702e-01	6.785e-01	1.950e-01
	FO	4.368e-01	1.703e+00	8.771e-01	9.387e-01	8.351e-02
20	NPO	9.732e-01	3.876e+00	2.744e+00	2.733e+00	6.217e-01
	PSO	9.494e-01	3.621e+00	3.202e+00	2.979e+00	6.406e-01
	FO	1.826e+00	5.000e+00	3.269e+00	3.261e+00	4.416e-01
21	NPO	3.000e+02	4.001e+02	4.001e+02	3.982e+02	1.389e+01
	PSO	2.000e+02	4.002e+02	4.002e+02	3.649e+02	7.370e+01
	FO	4.002e+02	4.002e+02	4.002e+02	4.002e+02	2.201e-14
22	NPO	1.250e+01	4.068e+02	1.682e+02	1.741e+02	9.757e+01
	PSO	2.879e+01	4.089e+02	2.098e+02	2.039e+02	1.028e+02
	FO	2.532e+01	1.424e+03	5.736e+02	5.582e+02	8.571e+04
23	NPO	3.999e+02	1.837e+03	1.090e+03	1.097e+03	3.703e-02
	PSO	2.271e+02	1.575e+03	8.299e+02	8.140e+02	2.908e+02
	FO	3.874e+01	1.993e+03	4.077e+02	4.849e+02	1.019e+05
24	NPO	1.106e+02	2.175e+02	2.079e+02	2.022e+02	2.309e+01
	PSO	2.001e+02	2.211e+02	2.098e+02	2.099e+02	5.524e+00
	FO	2.001e+02	2.166e+02	2.002e+02	2.022e+02	2.163e+01
25	NPO	1.149e+02	2.202e+02	2.059e+02	2.059e+02	1.409e+01
	PSO	1.205e+02	2.214e+02	2.101e+02	2.087e+02	1.349e+01
	FO	2.001e+02	2.111e+02	2.002e+02	2.009e+02	5.615e+00
26	NPO	1.039e+02	2.000e+02	2.000e+02	1.736e+02	3.938e+01
	PSO	1.060e+02	3.176e+02	2.000e+02	1.864e+02	7.060e+01
	FO	1.020e+02	3.144e+02	2.002e+02	1.969e+02	8.862e+03
27	NPO	3.002e+02	5.414e+02	3.059e+02	3.409e+02	6.418e+01
	PSO	3.003e+02	6.391e+02	4.935e+02	4.373e+02	1.224e+02
	FO	3.010e+02	4.000e+02	4.000e+02	3.559e+02	2.416e+03
28	NPO	1.000e+02	6.426e+02	3.000e+02	3.423e+02	1.239e+02
	PSO	1.000e+02	3.000e+02	3.000e+02	2.882e+02	4.706e+01
	FO	3.001e+02	4.001e+02	3.001e+02	3.217e+02	1.727e+03

**Table 6-2:** Benchmark for NPO, PSO, FO: tested with 10-dimensional functions in CEC 2013 competition[74]

## 6.2 Tests in the high dimensional search space

Function	NPO	PSO	FO
1	1	1	3
2	2	3	1
3	3	2	1
4	1	2	3
5	2	1	3
6	1	2	3
7	2	3	1
8	1	2	3
9	3	2	1
10	3	2	1
11	1	2	3
12	2	3	1
13	3	2	1
14	1	2	3
15	3	2	1
16	2	3	1
17	2	1	3
18	2	3	1
19	1	2	3
20	1	2	3
21	2	1	3
22	1	2	3
23	3	2	1
24	1	3	1
25	2	3	1
26	1	2	3
27	1	3	2
28	3	1	2
Mean Rank	1.821	2.107	2.0

**Table 6-3:** Performance comparison of NPO, PSO, and FO in the mean ranking[74]



## 6.2 Tests in the high dimensional search space

Function	Best	Worst	Median	Mean	Variance
1	NPO	7.336e-06	1.546e-04	2.496e-05	3.212e-05
	PSO	5.634e-03	5.443e-05	5.531e-04	9.455e-04
	FO	4.030e-05	8.755e-03	2.590e-03	2.788e-03
2	NPO	3.496e+06	1.877e+07	1.219e+07	1.181e+07
	PSO	4.412e+07	5.542e+06	1.850e+07	1.955e+07
	FO	1.496e+07	3.682e+07	2.414e+07	2.445e+07
3	NPO	2.121e+07	1.520e+09	2.828e+08	4.378e+08
	PSO	1.678e+09	2.797e+07	2.771e+08	4.075e+08
	FO	9.015e+07	1.227e+10	4.713e+09	4.978e+09
4	NPO	4.726e+02	2.361e+03	1.064e+03	1.184e+03
	PSO	1.672e+04	7.799e+03	1.212e+04	1.203e+04
	FO	4.982e+04	8.882e+04	6.376e+04	6.349e+04
5	NPO	2.810e-03	1.200e-02	6.841e-03	7.019e-03
	PSO	7.281e-02	4.830e-03	2.260e-02	2.769e-02
	FO	6.943e+01	1.636e+02	1.155e+02	1.140e+02
6	NPO	5.557e+00	1.283e+02	7.353e+01	6.777e+01
	PSO	1.559e+02	2.002e+01	9.432e+01	9.291e+01
	FO	2.290e+01	1.190e+02	7.284e+01	7.183e+01
7	NPO	1.253e+01	8.567e+01	3.922e+01	3.971e+01
	PSO	7.794e+01	8.197e+00	4.031e+01	4.275e+01
	FO	4.720e+01	1.085e+02	7.209e+01	7.381e+01
8	NPO	2.084e+01	2.111e+01	2.097e+01	2.097e+01
	PSO	2.105e+01	2.083e+01	2.095e+01	2.095e+01
	FO	2.084e+01	2.106e+01	2.095e+01	2.095e+01
9	NPO	8.156e+00	4.121e+01	1.840e+01	1.885e+01
	PSO	3.541e+01	1.234e+01	2.217e+01	2.176e+01
	FO	1.329e+01	2.786e+01	2.068e+01	2.045e+01
10	NPO	2.015e+00	1.371e+01	5.153e+00	5.400e+00
	PSO	2.756e+01	1.240e+00	4.394e+00	5.273e+00
	FO	1.641e+00	1.755e+01	4.856e+00	5.509e+00
11	NPO	1.323e+01	4.481e+01	2.861e+01	2.872e+01
	PSO	5.561e+01	1.606e+01	3.011e+01	3.099e+01
	FO	3.384e+01	9.851e+01	6.570e+01	6.597e+01
12	NPO	3.331e+01	2.214e+02	7.170e+01	1.088e+02
	PSO	2.351e+02	3.360e+01	1.124e+02	1.179e+02
	FO	3.583e+01	9.752e+01	6.174e+01	6.085e+01
13	NPO	7.992e+01	2.123e+02	1.660e+02	1.617e+02
	PSO	2.511e+02	7.184e+01	1.969e+02	1.917e+02
	FO	1.034e+02	2.234e+02	1.514e+02	1.562e+02
14	NPO	4.715e+02	2.072e+03	9.579e+02	1.008e+03
	PSO	2.601e+03	5.521e+02	1.055e+03	1.102e+03
	FO	2.056e+03	4.692e+03	3.279e+03	3.263e+03
15	NPO	2.034e+03	8.073e+03	6.695e+03	5.359e+03
	PSO	7.7230e+03	5.015e+03	7.043e+03	6.945e+03
	FO	1.912e+03	4.827e+03	3.159e+03	3.163e+03
16	NPO	1.876e+00	3.269e+00	2.527e+00	2.554e+00
	PSO	2.775e+00	1.454e+00	2.285e+00	2.257e+00
	FO	1.095e-01	1.069e+00	3.377e-01	4.002e-01
17	NPO	5.644e+01	1.481e+02	7.926e+01	8.177e+01
	PSO	1.807e+02	7.090e+01	1.160e+02	1.174e+02
	FO	4.430e+01	8.163e+01	6.049e+01	6.102e+01
18	NPO	1.993e+02	2.571e+02	2.224e+02	2.235e+02
	PSO	3.177e+02	1.886e+02	2.613e+02	2.591e+02
	FO	5.570e+01	1.277e+02	7.408e+01	7.820e+01
19	NPO	2.299e+00	7.928e+00	4.107e+00	4.299e+00
	PSO	9.857e+00	2.809e+00	5.194e+00	5.395e+00
	FO	2.209e+00	7.152e+00	3.889e+00	4.042e+00
20	NPO	1.017e+01	1.500e+01	1.194e+01	1.200e+01
	PSO	1.500e+01	1.150e+01	1.500e+01	1.385e+01
	FO	1.451e+01	1.500e+01	1.500e+01	1.498e+01
21	NPO	2.001e+02	4.435e+02	3.001e+02	3.249e+02
	PSO	4.436e+02	1.028e+02	3.009e+02	3.525e+02
	FO	2.005e+02	6.408e+02	3.007e+02	3.307e+02
22	NPO	4.523e+02	1.828e+03	9.580e+02	1.020e+03
	PSO	1.896e+03	5.436e+02	1.174e+03	1.165e+03
	FO	2.974e+03	7.244e+03	4.991e+03	5.171e+03
23	NPO	2.073e+03	7.974e+03	4.425e+03	4.993e+03
	PSO	7.919e+03	5.916e+03	7.151e+03	7.121e+03
	FO	3.571e+03	6.590e+03	5.603e+03	5.453e+03
24	NPO	2.107e+02	2.633e+02	2.296e+02	2.316e+02
	PSO	2.924e+02	2.510e+02	2.685e+02	2.675e+02
	FO	2.085e+02	2.656e+02	2.341e+02	2.339e+02
25	NPO	2.551e+02	2.884e+02	2.721e+02	2.721e+02
	PSO	3.207e+02	2.627e+02	2.904e+02	2.900e+02
	FO	2.046e+02	3.040e+02	2.746e+02	2.667e+02
26	NPO	2.001e+02	3.521e+02	2.004e+02	2.192e+02
	PSO	3.776e+02	2.002e+02	2.014e+02	2.656e+02
	FO	2.002e+02	3.509e+02	3.305e+02	3.030e+02
27	NPO	4.456e+02	8.760e+02	7.468e+02	7.455e+02
	PSO	1.149e+03	7.335e+02	9.153e+02	9.164e+02
	FO	4.641e+02	8.727e+02	6.769e+02	6.698e+02
28	NPO	1.002e+02	1.437e+03	3.002e+02	3.186e+02
	PSO	1.531e+03	1.006e+02	3.015e+02	3.611e+02
	FO	1.008e+02	2.545e+03	3.016e+02	3.886e+02

**Table 6-4:** Benchmark for NPO, PSO, FO: tested with 30-dimensional functions in CEC 2013 competition

## 6.2 Tests in the high dimensional search space

Function	NPO	PSO	FO
1	1	2	3
2	1	2	3
3	2	1	3
4	1	2	3
5	1	2	3
6	1	3	2
7	1	2	3
8	3	1	2
9	1	3	2
10	2	1	3
11	1	2	3
12	2	3	1
13	2	3	1
14	1	2	3
15	2	3	1
16	3	2	1
17	2	3	1
18	2	3	1
19	2	3	1
20	1	2	3
21	1	3	2
22	1	2	3
23	1	3	2
24	1	3	2
25	2	3	1
26	1	2	3
27	2	3	1
28	1	2	3
Mean Rank	1.5	2.357	2.143

**Table 6-5:** Performance comparison of NPO, PSO, and FO in the mean ranking



## 7

# Conclusion

NPO takes a deterministic approach based on a well-established mathematical operation, Newton method. Due to the inherent fractal nature and strong convergence of the method, NPO seems to enjoy both features of exploration and exploitation, making effective optimizations for a wide range of functions.

For NPO, the balance between exploration and exploitation can be handled simply as a property of a multi-dimensional mapping by a guiding function. Because such mapping can be easily created by conditions of an ideal guiding function in §5.1, it can be analysed and tuned for the convergent/divergent movements of searching agents. This characteristic provides us with consideration of customizing the guiding function depending on the quality of test functions for future work.

For local convergence of NPO, we partly adopted randomness in NPO, but this may not be necessary. There are some chaotic-based optimizers guaranteed local convergence without randomness[43]. Many metaheuristic methods including NPO drive the particles by the force derived from the interaction among population. This characteristic ends up with particles' stagnation and lack of diversity. Though NPO is one of them, it has inherent pseudo randomness using chaos unlike other methods. We can use this property to temporal best instead of randomness.

We suggested two ways to pick the leading particles in this dissertation but there can be proposed more ways. Choosing leading particles is a parameter to control the balance between exploration and exploitation in NPO. Especially suggested ways consider the fitness values of the particles only. We can use other information such as geographic information of leading particles or the multiplicity of a guiding function. We can choose the distant leading particles. This may get worse the performance because of its poor exploitation ability but this is good at exploration. Moreover, it is known that if the derivative of a function is closely zero, then the particle runs away from the root by Newton method. The balance between exploration and exploitation can be controlled by leading particles and guiding functions.

Besides, there are many general problems in global optimization. As we introduced some technical problems in §1.2.3, it is important to set the proper criteria in global optimizer such as stopping criteria, initialization, the number of fitness evaluation, etc. Not much research has been done yet in spite of its importance. To evaluate the performance of an optimizer, we need the adaptable measure. Mean ranking is dependent on the fitness value, which is not desired. Metaheuristics consider the universal structure of fitness function regardless of its specific characteristics or scale. And for practical use, we suffer the curse of dimensionality of search space. The search space is exponentially increased as the dimension gets higher. This difficulty should be handled in proper way practically. What is more, the ideal balance between exploration and exploitation is unknown. Though we handle this by trial and error, there need more rigorous research to clear the global optimization problem from this respect.

# References

- [1] Abido, M. (2002). "Optimal design of power-system stabilizers using particle swarm optimization." *IEEE transactions on energy conversion* 17(3): 406-413. 12
- [2] Abraham, A., et al. (2008). "Swarm intelligence algorithms for data clustering. *Soft computing for knowledge discovery and data mining.*" Springer: 279-313. 12
- [3] Beni, G. (2004). "From swarm intelligence to swarm robotics." *International Workshop on Swarm Robotics*, Springer. 9
- [4] Bonyadi, M. R., et al. (2014). "Particle swarm optimization for single objective continuous space problems: a review." *Evolutionary Computation* 1530: 9304.
- [5] Chou, C.-W., et al. (2013). "Markov Chain and Adaptive Parameter Selection on Particle Swarm Optimizer." *International Journal on Soft Computing* 4(2): 1. 42, 48
- [6] Clerc, M. and J. Kennedy (2002). "The particle swarm-explosion, stability, and convergence in a multidimensional complex space." *IEEE transactions on evolutionary computation* 6(1): 58-73. 10, 40
- [7] Cohen, A., et al. (2018). "Diverse Exploration for Fast and Safe Policy Improvement." *arXiv preprint arXiv:1802.08331*. 5
- [8] Dorigo, M. (1992). "Optimization, learning and natural algorithms." PhD Thesis, Politecnico di Milano. 1, 10
- [9] Du, K.-L. and M. Swamy (2016). "Search and optimization by metaheuristics." Birkhauser July. 15
- [10] Du, K.-L. and M. N. Swamy (2013). "Neural networks and statistical learning." Springer Science Business Media. 13
- [11] Eberhart, R. and J. Kennedy (1995). "A new optimizer using particle swarm theory." *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on, IEEE*. 1, 9

- 
- [12] Emary, E. and H. M. Zawbaa (2016). "Impact of chaos functions on modern swarm optimizers." *PloS one* 11(7): e0158738. 2, 15
- [13] Engelbrecht, A. P., et al. (1999). "Training product unit neural networks." 13
- [14] Fister, I., et al. (2013). "A comprehensive review of firefly algorithms." *Swarm and Evolutionary Computation* 13: 34-46.
- [15] Fister Jr, I., et al. (2015). "A review of chaos-based firefly algorithms: perspectives and research challenges." *Applied Mathematics and Computation* 252: 155-165. 2, 15
- [16] Gaing, Z.-L. (2004). "A particle swarm optimization approach for optimum design of PID controller in AVR system." *IEEE transactions on energy conversion* 19(2): 384-391. 12
- [17] Gao, S., et al. (2006). "Convergence analysis of particle swarm optimization algorithm." *Science Technology and Engineering* 6(12): 1625-1627. 10
- [18] Ghalia, M. B. (2008). "Particle swarm optimization with an improved exploration-exploitation balance." *Circuits and Systems, 2008. MWSCAS 2008. 51st Midwest Symposium on, IEEE*. 40
- [19] Glover, F. (1986). "Future paths for integer programming and links to artificial intelligence." *Computers operations research* 13(5): 533-549. 1, 8
- [20] Gordon, D. M. (2010). "Ant encounters: interaction networks and colony behavior." Princeton University Press.
- [21] Guerra, F. A. and L. d. S. Coelho (2008). "Multi-step ahead nonlinear identification of Lorenzs chaotic system using radial basis neural network with learning by clustering and particle swarm optimization." *Chaos, Solitons Fractals* 35(5): 967-979. 42, 48
- [22] Guo, C., et al. (2009). "Optimal control of continuous annealing process using PSO." *Automation and Logistics, 2009. ICAL'09. IEEE International Conference on, IEEE*. 12
- [23] Gupta, A. K., et al. (2006). "The interplay between exploration and exploitation." *Academy of management journal* 49(4): 693-706. 5
- [24] Hassanzadeh, I. and S. Mobayen (2007). "Optimum design of PID controller for 5-bar-linkage manipulator using particle swarm optimization." *Proceeding of the 4th t International Symposium on Mechatronics and its Applications (ISMA07), Sharjah, UAE March*. 12
- [25] He, Q., et al. (2007). "Parameter estimation for chaotic systems by particle swarm optimization." *Chaos, Solitons Fractals* 34(2): 654-661. 2

- [26] Hilborn, R. C. (2000). "Chaos and nonlinear dynamics: an introduction for scientists and engineers." Oxford University Press on Demand. 15, 16
- [27] Holland, J. H. (1992). "Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence." MIT press. 11
- [28] Hubbard, J., et al. (2001). "How to find all roots of complex polynomials by Newtons method." *Inventiones mathematicae* 146(1): 1-33. 16
- [29] Kennedy, J. (2000). "Stereotyping: Improving particle swarm performance with cluster analysis." *Evolutionary Computation*, 2000. Proceedings of the 2000 Congress on, IEEE. 12
- [30] Khodier, M., et al. (2008). "Design of multi-band multi-section transmission line transformer using particle swarm optimization." *Electrical Engineering* 90(4): 293-300. 42, 48
- [31] Li, M., et al. (2013). "Prediction of gas solubility in polymers by back propagation artificial neural network based on self-adaptive particle swarm optimization algorithm and chaos theory." *Fluid Phase Equilibria* 356: 11-17. 2, 15
- [32] Li, N., et al. (2006). "An analysis for a particle's trajectory of PSO based on difference equation." *Jisuanji Xuebao/Chinese Journal of Computers* 29(11): 2052-2061. 10
- [33] Liang, J., et al. (2013). "Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization." *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report 201212*: 3-18. 43
- [34] Liao, Q., et al. (2017). "Parameter estimation of nonlinear systems by dynamic cuckoo search." *Neural computation* 29(4): 1103-1123.
- [35] Liu, H.-b., et al. (2006). "Convergence analysis of particle swarm optimization and its improved algorithm based on chaos." *Control and decision* 21(6): 636. 10
- [36] Lu, Z.-S. and Z.-R. Hou (2004). "Particle swarm optimization with adaptive mutation." *Acta electronica sinica* 32(3): 416-420. 10
- [37] Malik, S. and S. Wadhwa (2014). "Preventing premature convergence in genetic algorithm using DGCA and elitist technique." *Int J Adv Res Comput Sci Soft Eng* 4(6). 16
- [38] Mandelbrot, B. B. (1982). "The fractal geometry of nature." WH freeman New York. 17
- [39] Mengxia, L., et al. (2016). "The Particle Swarm Optimization Algorithm with Adaptive Chaos Perturbation." *International Journal of Computers, Communications Control* 11(6). 2, 15

## REFERENCES

- 
- [40] Miller, P. (2010). "Smart swarm." HarperCollins UK.
  - [41] Mo, Y.-b., et al. (2013). "Optimal choice of parameters for firefly algorithm." Digital Manufacturing and Automation (ICDMA), 2013 Fourth International Conference on, IEEE. 48
  - [42] Molga, M. and C. Smutnicki (2005). "Test functions for optimization needs." Test functions for optimization needs 101. xi, 6, 42, 43
  - [43] Okamoto, T. and H. Hirata (2010). "Global optimization using a multi-point type quasi-chaotic optimization method with the simultaneous perturbation gradient approximation." Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on, IEEE. 16, 53
  - [44] ping Tian, D. (2013). "A review of convergence analysis of particle swarm optimization." International Journal of Grid and Distributed Computing 6(6): 117-128. 10
  - [45] Poli, R. (2008). "Analysis of the publications on the applications of particle swarm optimisation." J. Artif. Evol. App. 2008: 1-10. 12
  - [46] Poli, R., et al. (2007). "Particle swarm optimization." Swarm intelligence 1(1): 33-57.
  - [47] Rakitianskaia, A. and A. P. Engelbrecht (2009). "Training neural networks with PSO in dynamic environments." Evolutionary Computation, 2009. CEC'09. IEEE Congress on, IEEE. 13
  - [48] Ren, Z.-H., et al. (2011). "The global convergence analysis of particle swarm optimization algorithm based on Markov chain." Control Theory Applications 28(4): 462-466. 16
  - [49] Rickles, D., et al. (2007). "A simple guide to chaos and complexity." Journal of Epidemiology Community Health 61(11): 933-937. 15
  - [50] Rudenko, O. and M. Schoenauer (2004). "A steady performance stopping criterion for Pareto-based evolutionary algorithms." 6th International Multi-Objective Programming and Goal Programming Conference. 6
  - [51] Sauer, T. (2012). "Numerical Analysis (2nd)." Addison-Wesley, New Jersey, USA. 16
  - [52] Sharma, S. and G. P. Rangaiah (2013). "An improved multi-objective differential evolution with a termination criterion for optimizing chemical processes." Computers Chemical Engineering 56: 155-173. 6
  - [53] Sheng, Z., et al. (2014). "Parameter estimation for chaotic systems using a hybrid adaptive cuckoo search with simulated annealing algorithm." Chaos: An Interdisciplinary Journal of Nonlinear Science 24(1): 013133. 2

## REFERENCES

- 
- [54] Shi, Y. and R. C. Eberhart (1998). "Parameter selection in particle swarm optimization." International conference on evolutionary programming, Springer.
  - [55] Snaselova, P. and F. Zboril (2015). "Genetic algorithm using theory of chaos." *Procedia Computer Science* 51: 316-325. 2, 15
  - [56] Takeuchi, M., et al. (2016). "Firefly algorithm existing leader fireflies." *Circuits and Systems (APCCAS), 2016 IEEE Asia Pacific Conference on, IEEE.* 48
  - [57] Van Den Bergh, F. (2001). "An analysis of particle swarm optimizers." *University of Pretoria South Africa.* 10, 28, 29, 30, 31
  - [58] Van den Bergh, F. and A. P. Engelbrecht (2000). "Cooperative learning in neural networks using particle swarm optimizers." *South African Computer Journal* 2000(26): 84-90.
  - [59] Van der Merwe, D. and A. P. Engelbrecht (2003). "Data clustering using particle swarm optimization." *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on, IEEE.* 12
  - [60] Van Veldhuizen, D. A. and G. B. Lamont (2000). "On measuring multiobjective evolutionary algorithm performance." *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on, IEEE.* 6
  - [61] Weise, T. (2009). "Global optimization algorithms-theory and application." Self-published 2. 1, 4
  - [62] Wolpert, D. H. and W. G. Macready (1997). "No free lunch theorems for optimization." *IEEE transactions on evolutionary computation* 1(1): 67-82. 3, 5, 41
  - [63] Wong, J. Y., et al. (2016). "Design of shell-and-tube heat exchangers for multiple objectives using elitist non-dominated sorting genetic algorithm with termination criteria." *Applied Thermal Engineering* 93: 888-899.
  - [64] Wu, X. and Z. Chen (1996). "Introduction of chaos theory." *Shanghai Science and Technology, Bibliographic Publishing House.* 14
  - [65] Yang, X.-S. (2009). "Firefly algorithms for multimodal optimization." *International symposium on stochastic algorithms, Springer.* 1, 11
  - [66] Yang, X.-S. (2017). "Nature-inspired algorithms and applied optimization." *Springer.* 11
  - [67] Yang, X.-S. (2018). "Mathematical Analysis of Nature-Inspired Algorithms." *Nature-Inspired Algorithms and Applied Optimization, Springer: 1-25.* 4
  - [68] Yuan, X., et al. (2015). "Parallel chaos optimization algorithm with migration and merging operation." *Applied Soft Computing* 35: 591-604.

## REFERENCES

- 
- [69] Zhang, L.-p., et al. (2004). "Analysis and improvement of particle swarm optimization algorithm." *Information and control* 33(5): 513-517. 10
  - [70] Zhang, Z.-H., et al. (2015). "Parameter estimation of atmospheric refractivity from radar clutter using the particle swarm optimization via Levy flight." *Journal of Applied Remote Sensing* 9(1): 095998.
  - [71] Zitzler, E. and L. Thiele (1998). "Multiobjective optimization using evolutionary algorithmsa comparative case study." *International conference on parallel problem solving from nature*, Springer. 6
  - [72] Kenneth Sorensen (2015). "Metaheuristicsthe metaphor exposed." *International Transactions in Operational Research*, 22(1):3-18, 2015. 3
  - [73] Yang, X.-S. (2011). "Engineering Optimization : An Introduction with Metaheuristic Applications." Wiley. 9
  - [74] Jeong, S. and Kim, P (2019). "A population based optimization method using Newton fractal." *Complexity, in press*. ix, x, xi, 2, 7, 18, 19, 21, 22, 24, 26, 33, 34, 39, 40, 49, 50



## 감사의 글

먼저 7년동안 정성으로 가르쳐주신 김필원 교수님께 말로 표현하기에는 부족한 큰 은혜에 감사드립니다. 연구자로서, 스승으로서, 존경스럽다고 전하고 싶다.

박사 학위논문 심사를 맡아주신 현윤경 박사님, 이창형 교수님, 장봉수 교수님, 최진혁 교수님께도 감사드립니다. 특히 인턴 기간동안 세심하게 신경 써주시고, 연구와 삶에 대해서 값진 조언을 해주신 현윤경 박사님께 감사드립니다.

내가 가장 좋아하는 과목인 해석학을 열정적으로 가르쳐주신 권봉석 교수님께 만나서 너무 다행이었다고 말씀드리고 싶다. 또 힘들어서 찾아볼 때마다 필요한 조언을 해주시는 신상묵 교수님께 연구자의 마음가짐을 배울 수 있어서 감사의 인사를 드리고 싶다. 늘 반갑게 맞아주시고, 시간을 내어서 조언을 해주신 배한택 교수님, 허인조 교수님께 감사드리고 싶다. 인생의 선배로서 등대같은 역할을 해주셨던 김형욱 선생님, 한광석 선생님, 이지혜 선생님께 감사드리고 싶다.

대학원에 재학하는 동안 항상 나를 지지해 준 Ng Thien Binh에게 진심으로 고맙다.(I really appreciate for your support, Binh.) 나를 배려해주고, 도와준 선후배님들(최준호, 배준식, 이준영, 김현기, 김은지, 심완용, 박종오, 이세연, 강태웅, 김학규, 윤주동, 윤홍규, 최재성, 이우제, 최현영, 이효정 박사님)께 고마웠다는 인사를 하고 싶다. 피곤한 몸으로 기숙사에 돌아와서 밤 늦게까지 이야기를 들어준 나의 룸메들, 윤정하, 김선이 박사님, 최기림에게 고맙다.

짧지 않은 인턴 기간동안 많이 배려해주신 고태욱 박사님, 이승희 박사님, 이종걸 박사님께도 감사드립니다.

부족한 나를 아껴주고 친구로서 사랑해주신 김민희, 황지현, 신경희에게 영원히 함께 하고 싶다고 전하고 싶다.

마지막으로 소중한 부모님(정동호, 방정숙), 언니 정주영, 남동생 정현성에게 언제나 사랑한다고 전하고 싶다.